End Device and Network Adaptation of WaveVideo Streams

Christian Kücherer, Andreas Schrader, Andreas Kassler, Oliver Haase

Abstract— **Increasing heterogeneity in terms of network and device capabilities, as well as user specific QoS profiles require media adaptation in potentially all nodes of a transmission path. Consequently, there is a need for a comprehensive end-to-end resource management to provide predictable QoS to future wireless multimedia systems.**

We propose syntactical and semantical filter algorithms to provide end-to-end adaptation of layered WaveVideo streams to be integrated with the MASA QoS framework, which features QoS management and negotiation facilities.

We introduce syntactical and semantical filters, which can operate at the sender, in inner network nodes, and at the receiver side. By evaluation studies we demonstrate the effectiveness of our approach.

Keywords— **Quality of Service, Video Adaptation, Video Streaming, Video Filtering, Java, JMF, WaveVideo.**

I. Introduction

ONE of todays most challenging research areas in telecommunications is the provision of mobility in very different regards. *Terminal mobility* allows for personal movements while using a wireless device, *user mobility* supports some kind of virtual home environment by using different devices, *session mobility* allows a user to switch devices even during ongoing (multimedia) sessions, etc. As a second major trend, the increasing bandwidth support of evolving wireless technologies in future 3G or 4G communication networks allows for new and enhanced multimedia streaming applications, such as real time video conferencing, multimedia distance learning, video on demand, and many more.

The most challenging problem to cope with in such an environment is the high level of heterogeneity, which is imposed by the large number of different access technologies (especially in the wireless case) and the growing variety of user devices with very different capabilities. In addition, the increasing bandwidth-consumption of multimedia applications more and more often leads to congestion situations in the network. Since the current Internet does not support resource reservation or QoS guarantees, applications are faced with fluctuating network transmission and device performance characteristics. Especially for a scenario, where a mobile terminal performs handoffs between different access technologies, the variation of end to end performance will be extremely dynamic.

C. Kücherer and A. Schrader are with NEC Europe Ltd., Network Laboratories, Heidelberg, Germany. E-mail: Christian.Kuecherer|Andreas.Schrader@ccrle.nec.de.

A. Kassler is with the Distributed Systems Department, University of Ulm, Ulm, Germany. E-mail: kassler@informatik.uni-ulm.de.

O. Haase is with the Networking Techniques Research Department, Bell Labs Research, Holmdel, NJ, USA E-mail: oli@dnrc.bell-labs.com

For the development of future communication scenarios, taking the variability of all these parameters into account, an adaptive transmission mechanism provides benefits, which allows for a high flexibility. Adaptation can take place at the sender, at the receiver and at intermediate nodes in the transmission path. Sender rate adaptation offers the highest flexibility, since the full semantical information of the data source is available. Especially in multicast sessions, dedicated media adaptation units (also referred to as network filter units) can adapt the stream to the aggregated demands of the following subtree. Filtering at the receiver can be useful, if limited local resources are available. With this combination, a long-term, slow adaptation to the static demands of the session can be achieved by using appropriate feedback mechanisms. To deal with highly dynamic changes during the transmission, also simple filtering mechanisms inside routers can be used to allow for short-term, very fast, locally optimized reactions and seamless handoff performance on mobile clients. The behaviour of the overall adaptation strategy should be controllable by user QoS policies, using parameters like color resolution, frame size or frame rate, etc.

In order to allow a meaningful adaptation of the media stream, comprehensive monitoring must be used. Transmission parameters, like loss rate, delay, and jitter must be combined with system specific parameters, like CPU utilization, memory usage, codec internal delay, router queue fill grade, etc. Together with the user QoS policy, these values can be fed into a trader algorithm, which redefines the filter settings and thus influences resource usage. Since these parameters will be only available on specific hosts or links, a comprehensive QoS framework is necessary, which allows for collecting and distributing monitoring information. We use the MASA QoS framework for this purpose [1]. By using a distributed set of QoS Brokers, which are supported by pluggable QoS Manager modules, MASA is able to interactively coordinate mobility and media management.

In this paper we propose two different video adaptation strategies based on pure syntactical information for a simple, but efficient filter as well as a more complex semantical filter, allowing for highest flexibility. The paper is organized as follows. In the next section, we present the usefulness of media adaptation and analyze its problems. In section III, we give an overview of the MASA QoS Framework. In section IV, we give a brief introduction into the WaveVideo codec and describe the use of this codec for RTP streaming. Our architecture consisting of de-/packetizer and filter modules is outlined in section V. In section VI we present a set of syntactical and semanti-

cal filter algorithms for WaveVideo streams. Section VII reports on some first measurements performed for the syntactical and semantical filters. Finally, we conclude our paper and present ideas for future research.

II. Media Filtering - Foundations

Several adaptive codecs have been proposed (e.g. layered DCT [2], *lH.261* [3]), but they often suffered from the flexibility for the support of different receiver requirements. Recently, Wavelet based coding mechanisms have become very popular. In this paper we are using the WaveVideo codec [4], which offers a good compromise between compression ratio, error tolerance, implementation efficiency, and the ability to support various filter operations. Based on a hierarchical coding scheme, WaveVideo is able to support prioritization of packets.

One possible way of adaptation is to perform a codec change during the transmission session. But this strategy imposes several severe problems. First of all, all codecs must be provided on every network node (e.g. downloadable plug-ins using active router technology as proposed in [5]). In addition, some codecs need to adjust themselves to the processed incoming stream in order to achieve high compression ratios. Therefore, changing to a codec with a lower average data rate could even increase the short-term data rate.

There are different possibilities to realize scaled media streaming. In the receiver driven layered multicast (RLM) approach, proposed in [6], receivers can decide about the quality of the stream by joining or leaving the respective multicast groups, which contain different layers of information. But it has turned out, that this scheme has a lot of drawbacks. The most important problem of fixed layer subscription schemes is the unability to support different receiver QoS policies at the same time without inserting redundant layers.

For example, the number of multicast groups is restricted in many operating systems, thus preventing the provisioning of a high number of channels, which is necessary to provide detailed graduation. Also, the management of a large number of channels results in an immense processing overhead (e.g. changing routing tables). Since IGMP is used for the feedback of join and leave actions, no fast adaptation could be performed. But the most important drawback of streaming media information in layered multicast groups is, that this scheme does not support heterogeneous user QoS policies.

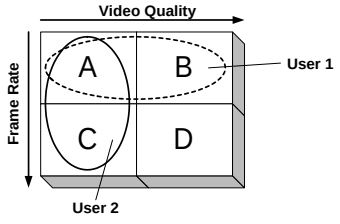

Fig. 1. Conflicting user QoS policies.

Lets for example assume, that User 1 wants to have the highest possible video quality and does not care about the frame rate. Instead, user 2 would like to maintain the full frame rate, independent of the quality of each frame(see Fig. 1). It is a hard question, how layered information which supports both degradation policies in the case of congestion can be created. If the layers contain hierarchical refinements of video quality, user 2 can't fulfil his policy and vice versa. For providing support to both users with their different policies, redundant transmission will become necessary.

Therefore, we propose to use a single (multicast) stream which contains the aggregated sum of all requested data parts for the respective receiver subtrees. Adaptation to the current group requirements is performed by a distributed set of filter nodes, exchanging feedback information. This allows for slow and mid-term adjustments of the multicast stream. To allow also very fast, short-term adaptation to deal with highly dynamic congestion situations, simple filter operations at network routers should be used, additionally. This allows for locally optimized reactions, e.g. in case of queue buffer overflows. In our example above, in a perfect environment with sufficient bandwidth, the complete stream consisting of information parts A, B, C, and D can be transmitted to the subtree consisting of users 1 and 2. If there is congestion, D can be dropped first. If the available bandwidth further decreases, B and/or C can also be dropped. Such a scheme is not possible with multi-layered transmission when we have much more fine-grained scaling steps.

There are also different possibilities how and on which system layer filtering should be performed. Filtering on the physical or data link layer usually does not work properly, since no information about the content of packets is available and therefore no useful QoS support could be achieved. Even if some technologies allow for link layer prioritization, this mechanisms are restricted and can be seen as special isolated parts of the Internet. The lowest layer in which priorities could be considered is the network layer.

By using the TOS (Type of Service) field of the IP header, differentiated treatment of packets could be realized. For example, in the IETF DiffServ architecture [7], the TOS field is used to distinguish packet priority levels. But this limited set of priorities only allows for a relatively static approach, with policies like "*Video has higher priority then HTTP traffic*". Such policies are usually defined with mid-term time constraints (days, months) and do not support differentiated treatment of users. If the marking process is directly performed at the sender system, a mapping between the importance of each packet to the correct DiffServ class would allow for a different treatment of each packet. Since DiffServ is supported widely on current router implementations, this would enable a rapid deployment. But the limited number of supported classes of service (CoS) in current DiffServ implementations does not support a very flexible adaptation mechanism. If the receivers also have to be supported with different adaptation policies, a replication of packets with re-prioritization

would be necessary, which increases the overall traffic.

To specify a larger number of different priorities, an RTP extension header could be used. For example, an RTP filter router has been developed at NEC Heidelberg [3], which drops packets using integer priorities in an RTP extension header. This also allows for mapping semantical information to different priorities within one stream/application. This approach can be implemented very efficiently with table-lookup mechanisms within the kernel and therefore only adds insignificant performance overhead. The problem is, that we still have semantic mapping functions performed at the sender. This can be dynamically modified due to feedback information, but can't support different filter strategies (for different users) at the same time.

Arbitrary complex filter operations could be performed by interpreting the semantical information (payload) of compressed bit streams. This also allows for remapping priorities and differentiated treatment of each receiver. To support also changing QoS policies, the filter algorithm should be realized in an adaptable way. This can be implemented with well-defined signalling protocols or by supporting downloadable filter algorithms, e.g. using intelligent active routers (e.g. [8]). Of course it is arguable, whether active routers will ever be deployed widely in the Internet due to the efficiency and security problems, but on the other hand, such a mechanism would allow for the most flexible solution.

III. The MASA QoS Framework

The *MASA* QoS Framework (Mobility and Service Adaptation in Heterogeneous Mobile Networks) is a joint project of NEC Europe Ltd. Heidelberg, Siemens AG Munich and the University of Ulm [1]. The *MASA* framework was designed to fulfill the requirements of a *comprehensive integrated end-to-end QoS multimedia management system*, invoking all entities on the transmission path, like sender, network nodes, transcoding nodes, switches, routers, filter nodes, gateways and receivers. *MASA* allows the usage of underlying network layer QoS technologies (e.g. DiffServ, IntServ, MPLS, etc.), and also hides the complexity of these mechanisms from the applications. By controlling the complete communication infrastructure *MASA* is able to support QoS in a way which can neither be realized inside the applications nor with the underlying network QoS technologies alone.

The framework follows an object-oriented design and most of the components (except some operating system specific tasks) are implemented in Java. This allows for downloading plugable components from different parties. Through the usage of open interfaces a high degree of flexibility is achieved.

A. Architectural Overview

The *MASA* QoS architecture consists of a distributed set of autonomous QoS Brokers (See fig. 2) which can be placed on the (potentially mobile) end-system, on intermediate network nodes (e.g. router, switches) and on transcoding units (gateways). Each Broker is responsible for the brokerage between managers with very different tasks, like resource, network, media, monitoring, policy and mobility management.

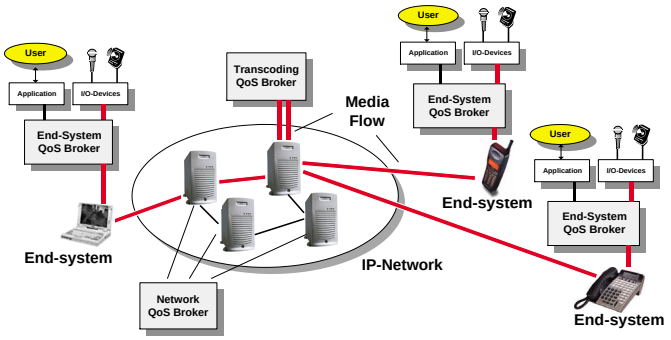

Fig. 2. Distributed set of autonomous QoS Brokers.

The main task of the *End-System QoS Broker* is to coordinate and manage local resources and provide adaptive media streaming services towards applications. Local and remote resources are orchestrated for multimedia streaming and service quality. User's QoS wishes are mapped to appropriate QoS parameters and terminal mobility is supported with the use of different access networks (e.g. GSM, WirelessLAN, UMTS, etc.), simultaneously.

Network QoS Brokers can be regarded as centralised QoS management units which support the end-system QoS Brokers and organise the orchestration of all streams in the respective network domain according to the network management policy. The network QoS Broker monitors network resources, decides admission in cooperation with other admission controlling entities (e.g. H.323 Gatekeepers) and realises load balancing and fairness concepts for all participating systems. The actual behaviour of the network QoS Broker depends on the location of the respective network node, e.g. core network router, access network router, media gateway, LAN switch etc.

The *Transcoding QoS Broker* can be used if heterogeneous clients must be supported in a multi-party conference scenario or if special network characteristics on certain network links have to be supported with mechanisms like adaptive Forward Error Correction (FEC), appropriate transmission protocols (e.g. wireless-TCP), etc. Transcoding QoS Brokers allow for automatic downloading of codecs on demand and also provide filter services.

Communication between the distributed collection of QoS Brokers is realized via appropriate interfaces. Main communication issues are capability exchange methods, QoS routing mechanisms, admission and authorization requests, and the management of media channels.

MASA presents applications with mechanisms for the processing and transmission of 'high quality' multimedia streams, i.e. adapted to the user's QoS wishes and the available infrastructure. Applications subscribe to the system and use the provided multimedia facilities via a respective QoS-API. The set-up of a complete chain of media processors, consisting of capture devices, codecs, effect processors, etc., can be controlled via this API. Appropriate

graphical user interfaces are provided to present media information (e.g. video panel). Since *MASA* provides a flexible mechanism to *plug-in* arbitrary components to capture, process, code, transmit, receive, decode and display any kind of media, the applications are shielded from that low-level complexity. Adaptive and layered coding can be used seamlessly by any application to allow for scalable media transmission.

With respect to mobility, efficient handoff algorithms are a cost-effective way of enhancing the capacity and QoS of cellular systems [9]. *MASA* supports mobile devices by integrating mobility management into the framework and using fast QoS re-negotiation and adaptation mechanisms to allow seamless intra- and inter-technology handoff.

B. Hierarchical QoS Broker Structure

Fig. 3 outlines the typical structure of a *MASA* QoS module. The *QoS Broker* is the central, local intelligence unit which is supported by a set of *QoS Managers* which in turn are supported by appropriate *QoS Controllers*. With this hierarchy of Manager/Controller structures, the QoS Broker delegates separate tasks for controlling and processing media streams and therefore provides a clear separation of tasks with different time constraints.

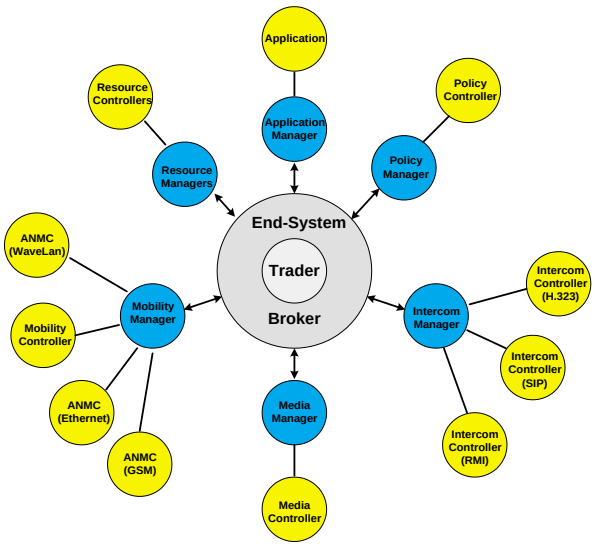

Fig. 3. Hierarchical structure of *MASA* on end-systems.

The *Policy Manager*, e.g., is responsible for the storage and retrieval of QoS preferences within a user profile and for presenting an appropriate policy GUI to the user. The Policy Controller enables the access to a policy database. The *Resource Managers* are responsible for controlling the available resources (like CPU, memory, network, etc.) via the respective Resource Controllers. The *Media Manager* is responsible for the provision and orchestration of actual media processing entities, like codecs, packetizers, etc., inside the Media Controller. It monitors the transmission parameters and reports aggregated statistic information to the Broker. It also implements the media adaptation mechanisms (like the filter operations described in section VI) as instructed by the Broker. The *Intercom Manager* is used to allow inter-Broker communication. The *Application Manager* provides mapping functionality between different categories of applications, like VoD or IP-Telephony and the Broker QoS API. The *Mobility Management* is responsible for the support of device mobility and enables the usage of different access devices. We have implemented a Mobility Manager using Mobile IP [10]. The Mobility Controller is a standard Mobile IP daemon. For each network interface an Access Network Monitoring Controller (ANMC) measures link quality parameters and reports them to the Mobility Manager, which either directly forces a handoff or informs the QoS Broker about available network options.

The tight integration of mobility management and handoff control within the QoS architecture allows for supporting seamless handoffs, fast adaptation (e.g. by changing filter settings, see section VI) as well as QoS-controlled handoff decisions.

The Broker regularly requests monitoring information from the managers. The aggregated monitoring information together with the user's QoS policy is used as input for the included trader mechanims which analyses the current situation and decides for possible adaptation of the current active sessions. On the end-system, the algorithm of the trader is controlled by a local trading policy which can be easily downloaded and exchanged even during runtime. The Broker parses the result and informs the respective managers about the necessary actions that have to be performed to realize the adaptation. Examples are codec changes and instantiation of filter operations inside the Media Manager or handoffs inside the Mobility Manager.

Since not all Managers have to be used for each Broker type, our design provides scalability. For example, at transcoding/filtering nodes, the Broker does not need Application or Mobility Managers.

Through the use of open interfaces between the QoS Brokers and Managers, the system can be easily extended with new Manager/Controller pairs. For example, if battery management should be included in the architecture, only a resource manager/controller for the battery needs to be developed and a new trading policy allows then to include power management issues into the codec selection process (like *if the battery power is low change to a simple codec*).

IV. WaveVideo over RTP

When transmitting media streams over networks, distributed multimedia applications first compress raw media data, then the compressed samples are packetized at the sender according to the compression format and finally the compressed and packetized samples are transmitted over the network using a transport protocol. On the receiver side, the packets are re-assembled into compressed media samples, decompressed and rendered.

A. RTP Multimedia Transmission

The Real-Time Transport Protocol (RTP) [11] is the protocol of choice when transmitting media streams over IP based networks. RTP together with its control protocol

RTCP are nowadays mainly included in the application itself. Applications implement the packetization functionality and generate proper RTP packets (i.e. the RTP payload format) that are then transmitted using standard socket calls. This requires each application to implement the same functionality: generating RTP packets out of compressed media samples and re-assembling the samples at the receiver. This packetization and re-assembling depends on the media codec and some RTP payload format definitions are standardized in a number of IETF RFCs. In the MASA framework, this work is performed within the Media Controller, allowing to develop multimedia streaming applications with significantly reduced implementation effort.

B. The WaveVideo codec

The WaveVideo codec is a very robust and efficient video coding algorithm, developed at the ETH Zürich [4]. It is a lossy compression algorithm, based on Wavelet transformation and temporal redundancy elimination. The codec is actually available as a native codec for Windows32 [1] and Linux platforms, but it can be expected that it will also be available for other platforms.

The coder generates a set of transformed and quantized Wavelet coefficients, which are decoded to an empty Wavelet tree at the decoder. Whenever this Wavelet tree is completed or a certain deadline is reached, this tree is processed and a YC_bC_rimage is created. The quality of the picture depends on the completeness of the Wavelet tree, the quantization thresholds at the sender and the network packet loss rate.

Input for the coder is a video signal in the RGB or in the YC_bC_r-colorspace which then is coded to the output stream. This output stream contains the transformed and quantized coefficients for the inverse wavelet transformation. Based on the logical structure of the wavelet tree, a certain number of the coefficients are packed together to a *network packet* and tags and headers, describing the semantics of each packet are added. These WaveVideo packets have small sizes and can be sent over the network using an arbitrary transport protocol.

C. Packetization of WaveVideo

As the WaveVideo codec is not standardized, it is required to build a packetization scheme for the use of WaveVideo for RTP based transmission. This packetization scheme should provide support for different kind of filter operations. In particular, our goal was to specify a RTP payload format that allows for efficient filtering of WaveVideo streams in network routers that inspect a dedicated field following the standard RTP header [3]. Another goal was to efficiently enable frame rate based filter services based on RTP header extensions. A special RTP-based router is thus able to decide based on its load information and its dropping policy to drop or forward a packet that belongs to a layered RTP flow by a simple look-up operation that indicates the layer number of the actual packet. The layer number can thus be seen as some kind of relative priority of the actual packet. In addition, the payload format has to enable the receiver to re-assemble compressed frames as network re-ordering may happen at any time.

[1]For the Windows32 platform, the codec is implemented as a dynamic link library (DLL).

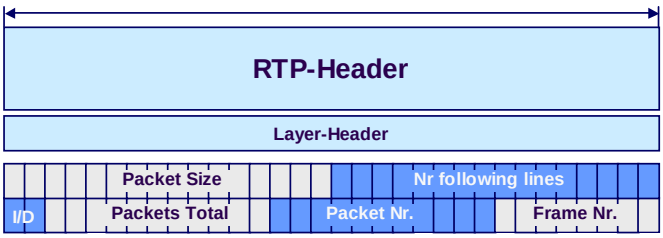

Fig. 4. The RTP-Layer header format provides support for RTP based media adaptation for layered video.

Fig. 4 shows the layout of the fields that follow the standard RTP header for the transmission of WaveVideo. First, the layer header (32 bits) can be used in network routers for simple and fast filtering. It contains a layer number (8 bits), a reserved extension field (8 bits) and a 16 bit sequence number per layer. Thus the receiver has knowledge about any lost or corrupted packets per layer. A mapping is required to derive the layer number from the semantical information available in the WaveVideo tag, in order to decide, which information is most important. The most important information has to be mapped to layer 0, the least important to layer 255. This mapping depends on the priorities and preferences of the user for the given session.

The WaveVideo specific packetizer header follows the layer header and contains the size of the packet, and a number of following lines which is used for future extensions. The *I/D* field allows to distinguish between I- and Delta frames because a special I-frame header is inserted in the compressed data if the packet belongs to an I-frame [4]. The *PacketsTotal* field indicates the total number of packets for the given frame. This information can be used for detecting, if all packets for the actual frame have been received and allow for selective re–transmit mechanisms. The *FrameNr* field indicates the frame number to which the packet belongs to. By including the *FrameNr* and *I/D* field, frame rate filter services can be implemented by RTP-based routers that simply inspect those two fields. Including the size of the packet, the packet number and the total number of packets enables the filter to scale the bitrate to a given target rate and to restrict the percentage of packets of a given frame that are to be forwarded.

For further details of the WaveVideo codec and the WaveVideo tag which enables semantical filter operation and which follows the RTP layer header and the packetizer header, see [4].

V. Architecture

In this section we introduce the design of our Media Controller implementations. Since our implementation is based on Java and JMF, we will start with a short overview of these technologies in subsection V-A. In subsection V-

B the internal structure of the Media Controllers for the sender, the receiver and for network filter nodes is presented. Finally, a detailed description of the inner components of the Media Controllers (packetizer V-C, depacketizer V-D, and filter V-E) is given.

A. Java and JMF

Java is an object oriented and platform independent programming language by Sun Microsystems. It is the chosen platform for future handheld devices and offers great possibilities for future applications. Since Java itself is not able to handle realtime based information like audio and video, Sun released the Java Media Framework (JMF) [12]. JMF is an extension package for Java and enables the handling, manipulation and transmission of realtime based data. Beside others, JMF offers also an RTP stack to transmit and receive media streams over networks. JMF supports different video codecs like MPEG, JPEG and H.261 [12]. To be able to support new or proprietary codecs, JMF offers a plug-in architecture to enable custom packetizers, depacketizers and effects as needed for the transmission and media handling. Furthermore JMF is able to use the native codecs installed on the local machine. Ideally, every arbitrary codec can be supported by JMF with appropriated plug-ins.

B. Structure of Media Controllers

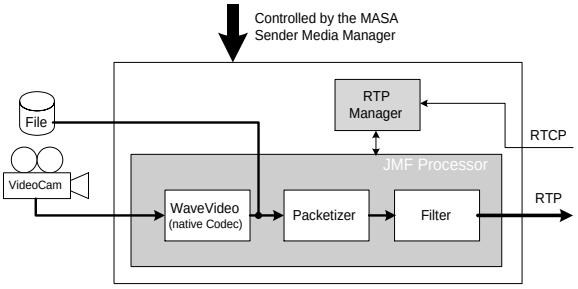

Fig. 5. The Sender Media Controller.

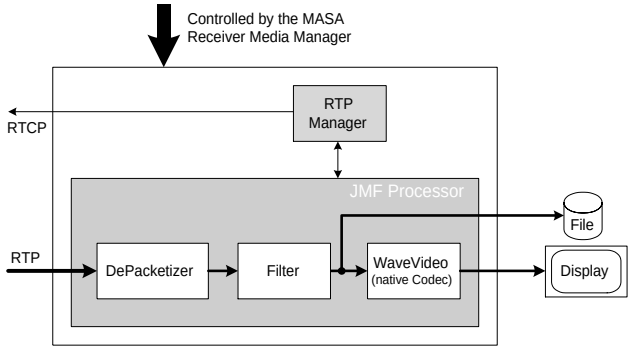

Fig. 6. The Receiver Media Controller.

To handle the transmission of WaveVideo streams, specialized Media Controllers have to be implemented and integrated in the MASA framework. Figure 5 and 6 outlines the structure of the Sender and Receiver Media Controllers.

Inside the sender Media Controller a filter can be used to achive long-term sender adaptation and to control the output stream of the sender. The filter inside the receiver Media Controller can be used to adjust the incoming stream to the available local resources. This can become necessary, if sudden resource restrictions appear for a short period of time (e.g. CPU overload, out of memory situations). The reasons to apply filters inside the network path is first, to support multicast scenarios with different user QoS policies, and second to allow for faster reaction to network congestion.

The filter settings of each entity are controlled with regard to network monitor information (latency, bandwidth, packet loss probability, etc.), end-system properties (CPU, memory) and user QoS policies (e.g. frame rate more important than color accuracy).

The processor of JMF is responsible to search apropriate plug-ins to convert the available input format to the selected output format. The chosen output format is RTP for the transmission over a network. If a WaveVideo file is read from disk, JMF does not need to apply the native codec, since the stream is already in the chosen format. Instead, if a capture device like a video camera is selected as a source, the signal has to be encoded to WaveVideo, thus the native codec is necessary. The packetizer plug-in enables JMF to process the WaveVideo stream in a way, that it is possible to send it over the network (see sectionV-C). The filter plug-in is used to adapt the stream to the requirements of the network, end-system and user. The structure of the filter is explained later in section V-E. The RTP manager, an abstract design component of JMF, is needed to stream the video data with RTP over a network and to control the RTP session, including the management of RTCP feedback information.

On the receiving side this process is quite similar. JMF builds up a processor, which is able to transform RTP data into a WaveVideo compliant stream. The decoder has to be applied, if it is necessary to display the video on a screen. The data is received, depacketized to obtain a stream and to identify separate frames, filtered if necessary and decoded to RGB or YC_bC_r, if necessary.

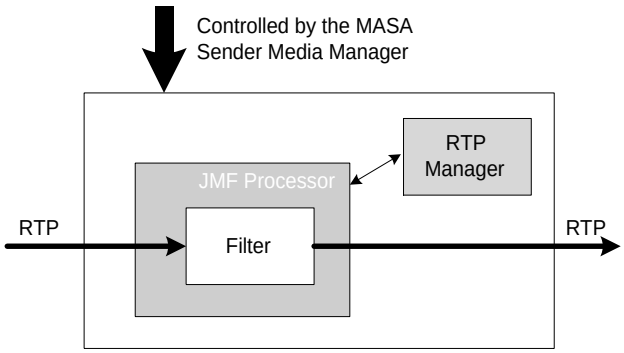

Fig. 7. The Network Filter Node Media Controller.

Figure 7 shows the structure of intermediate filter nodes somewhere in the network. The video stream is traced, and if the trader changes the quality requirements, the filter starts adapting the media stream by dropping certain packets.

C. Packetizer

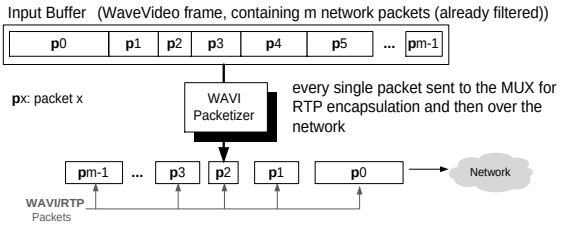

Fig. 8. Functionality of the packetizer.

The task of the packetizer is to extract WaveVideo packets out of a WaveVideo stream for the transmission over the network. Figure 8 outlines this mechanism. A complete WaveVideo encoded frame is given to the packetizer by JMF. Then the packetizer parses the frame and extracts every single network packet and gives it to the multiplexer. JMF will then take this data to create an RTP packet to sent it over the network. If every packet of the frame is sent, the packetizer requests JMF for the next frame and the fragmentation process will begin again.

The size of a WaveVideo frame does vary from frame to frame. Typical values for a video frame in full quality with the size of 384x288 pixels can be from 60 to 95 kbyte. To send the entire frame in one packet over the network would cause several risks. First of all, big packets have a higher risk to get lost in the network due to dropping policies on routers or other intermedia network nodes. Second, huge packets have a higer probability to get delivered too late. Thus, it is preferable to fragment the big frame into several smaller WaveVideo packets. Another advantage of the transmission of small packets is, that lost packets will only lead to a degradation of quality of the image, not to the loss of frames or even the complete stream.

D. Depacketizer

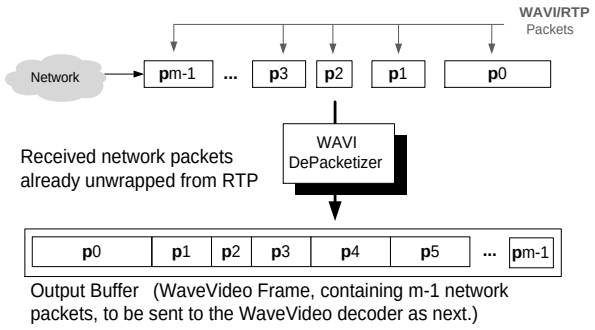

Fig. 9. Functionality of the depacketizer.

The receiver will receive these small WaveVideo packets that contain a certain quantization layer of a wavelet coefficient subtree. Figure 9 shows the strategy of the depacketizer. As much packets as possible belonging to one frame are collected and concatenated. The resulting frame is then passed to the next plug-in like the native WaveVideo decoder. The decoder constructs a wavelet tree out of the collected packets, and creates RGB or YC_bC_r images. If a certain deadline is reached or a packet of the next frame is received, the frame is considered to be complete and passed to the next plug-in.

E. Filter plug-in

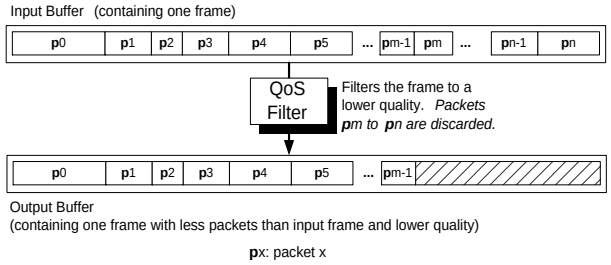

Fig. 10. Functionality of the QoS Filter plug-in.

The implemented JMF filter plug-in can be used to filter a WaveVideo stream in many respects. The algorithm, which is used for the filtering can be selected very easily. In our scenario we have implemented the filter in a way which allows the change of the filter-algorithm on the fly. The new algorithm will take effect in the next processed video frame. Examples for already implemented filter algorithms are

- random and priority mapped packet dropper and
- random and priority mapped bandwidth shaper,

which are described in more detail in the following section.

VI. Filter Algorithms

In this section we describe some example filter algorithms for WaveVideo codec video streams. Simple priority-based filtering based on syntactical information can be used to efficiently support a single adaptation policy (in either unicast or multicast scenarios). More complex structure-based filtering based on the semantical information of the WaveVideo Tag allows for different receiver adaptation policies. The more complex and semantic-based filter algorithms are described in [13].

A. Syntactical Filtering

The term syntactical filtering is used for algorithms, which base their packet dropping decisions on simple structure information, e.g. priority fields, or the packet's sequence number. This scheme can only support one single adaptation policy, since the priority fields can not reflect the relative importance of the packet for the respective users, but it can be applied to realize short-term adaptation and efficient still simple filtering.

A.1 Priority packet dropping

The WaveVideo coder produces several packets per video frame, containing the wavelet coefficients for the inverse wavelet transformation. One feature of the coder is, that it generates an already prioritized output. That means, that the important image information is carried by the first packet. Every more detailed and refinement information is carried by following packets. Since these packets have a sequence number in the WaveVideo header (see section IV), it is easy to assign the packets to baselayers and enhancement layers. Incoming packets, which are reordered, have to be ordered before filtering. The filter will let a certain number of packets pass, beginning from the baselayer. The number of packets m, which are allowed to pass is calculated as

$$m = trunc(n * q) \tag{1}$$

whereas q is the quality factor in $[0,1]$ and n is the number of packets of this frame. The first m packets of a frame ordered by priority will be passed and the least important packets ($packet_m..packet_n$) will be discarded.

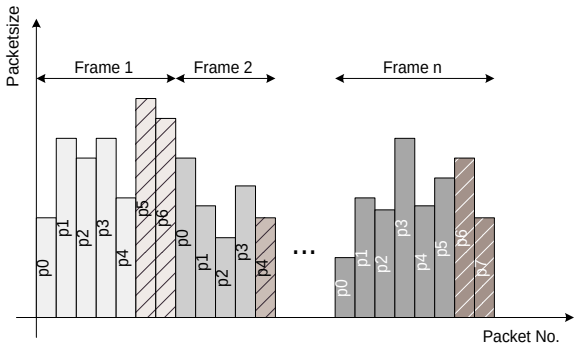

Fig. 11. Priority based packet dropper.

Fig. 11 presents a graphical representation. If we assume a quality factor of $q = 0.8$, then the filter will operate as follows: Frame 1 consists out of $n = 7$ packets (packet 0 to packet 6). With equation 1 m is calculated to $m = 5$, so 5 packets will pass (packet 0 to packet 4). The packets which are dropped are represented in the Figure as hatched bars. The next frame 2 consists out of $n = 5$ packets and we calculate $m = 4$, so packet 4 will be discarded. This scheme is applied to an arbitrary number of frames until q is changed.

The filter can be configured in such a way, that no more frames can pass. In case of $q = 0.0$, every packet of every frame will be discarded.

A.2 Priority data rate shaping

Since the sizes of the WaveVideo packets are not equally distributed within each frame, we also have implemented a filter used to adjust the video stream to the properties of the transmission channel in regard to available bandwidth. This filter allows q to have a direct effect on the produced data rate, whilst the q for the priority packet dropper only indirectly influences the data rate.

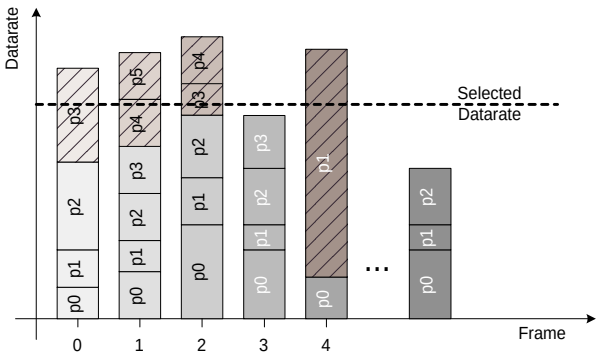

Fig. 12. Priority based data rate shaping.

Figure 12 shows the principle of the priority data rate shaping algorithm. Every vertical bar represents the packets of one frame and the hatched packets indicate, that this packet is discarded. The packets contain coefficients for the inverse wavelet transformation and the priorities are in correlation of their sequence number. Packet 0 is the one with the highest priority and following packets have decreasing priority. The data rate shaping algorithm considers every single frame and discards low priority packets. The first m packets of a frame will pass the filter, with m fulfilling the following restriction:

$$m <= numberOfPackets(frame_k) \tag{2}$$

and

$$\sum_{i=0}^{i=m} size(p_i) < datarate_{selected} \tag{3}$$

This will lead to a video stream with a roughly constant data rate, but some variations may be visible, since the packet sizes are not constant. In no case, the threshold value $datarate_{selected}$ will be exceeded. Again, if $datarate_{selected} = 0$ is selected, the filter will discard every packet of every frame.

B. Frame Rate Filter

Filters adapting the frame rate operate in the temporal domain and modify the frame rate thus reducing data rate (see Fig. 13). The frame number and the frame type (I- or Delta-frame) are coded within the RTP-packetizer header. The frame rate used by the source to emit the compressed frames is included in an I-frame header [4] which follows the RTP-packetizer header. Let us suppose that the sender sends at r_{src} and the receiver wants to be served with r_{dest}. A simple frame rate filter has been presented and evaluated in [14] and is based on the idea, that Filters need to know the frame type (Inter- or Intra-coded) and discard the frames according to their type while preserving r_{dest}. For each frame f_i to be processed, the next frame f_{fwd} is

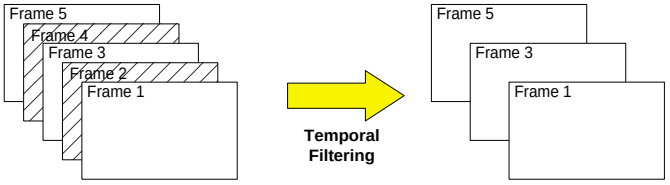

Fig. 13. The frame rate filter adapts the frame rate of a video stream

Fig. 14. The simple frame rate filter may produce annoying motion artefacts, if a Delta-frame refers to an I-frame that was dropped by the filter.

calculated which should not be dropped. If $r = \frac{r_{src}}{r_{dest}}$ denotes the ratio of source to target frame rate and frame f_i is forwarded, the next frame f_{fwd} to be forwarded is determined by adding r to f_i, i.e. $f_{fwd} = f_i + r$. All frames between f_i and f_{fwd} are dropped (i.e. all packets belonging to a dropped frame are dropped). Additionally, the frame rate field in the I-frame header is adjusted to match the new post filter frame rate and all Delta-frames referring to a dropped I-frame are dropped, too. Otherwise severe motion artefacts occur at the receiver as shown in Fig. 14. The simple frame rate filter for WaveVideo is a semantical filter. The filter works well, if the sequence contains only I-frames or if no Delta frames refer to a dropped I-frame.

In situations, where the source frame rate is not constant over time, this simple algorithm has also problems in estimating the source frame rate, which can be avoided by observing estimates of the source frame rate and the ratio of I-/Delta-frames. The *advanced frame rate filter* estimates the incoming frame rate r_{src} and the I-frame rate r^I_{src} using an AR(1) estimator.

$$r_{src} = \alpha \cdot r_{src} + (1-\alpha) \cdot \frac{r_{max}}{frameNr - lastFrameNr} \quad (4)$$

$$r^I_{src} = \beta \cdot r^I_{src} + (1-\alpha) \cdot \frac{r_{max}}{frameNr - lastIFrameNr} \quad (5)$$

where $\alpha, \beta < 1$. α determines the sensitivity of the estimator to fluctuating source frame rates. β determines the sensitivity to fluctuating I-frame rates. A number of test runs showed that a value of $\alpha = \beta = 0.8$ yields good results. The higher α, β the more sensitive the algorithm reacts to fluctuating source and I-frame rates. If $r^I_{src} > r_{dest}$ the actual I-frame is forwarded if the output frame rate does not exceed r_{dest}. All Delta-frames are filtered out. Otherwise, some Delta-frames have to be forwarded. In that case, a Delta-frame is only forwarded if the post filtered frame rate does not exceed r_{dest} when forwarding the next expected I-Frame. This algorithm assures, that for each forwarded Delta-frame the reference I-Frame is also forwarded. If that I-frame had been dropped, the decoder would associate a wrong I-frame with the Delta-frame thus producing artefacts similar to Fig. 14.

VII. Measurements

Within this section, we provide a preliminary performance evaluation of the filter approach based on WaveVideo syntactical and semantical filters. We were particular interested in post filter data rate versus visual quality of the stream. We conducted a number of test runs and applied the different filter algorithms. As video source file we used a digitized and WaveVideo compressed scene from the Beverly Hills Cop movie at 352x288 pixel resolution. The original frame rate was 25 fps, which yielded an average data rate of 1433 kbps. The total number of frames was 1518, the average number of successive Delta-frames following an I-frame was 3.1 (resulting in an average WGOP[2][15] size of 4.1). The maximum WGOP size was 5, the minimum was 1 and the standard deviation was 1.17. Maximum I-frame size was recorded to 17512 bytes, minimum I-frame size was 5608 bytes and average I-frame size was 10950 bytes. Maximum Delta-frame size was recorded to 14884 bytes, minimum Delta-frame size was 542 bytes and average Delta-frame size was 4905 bytes.

A. Evaluation of the priority mapped filter algorithms

In the following subsections we want to show some measurements to prove and evaluate the functioning of the priority packet dropping algorithm (VII-A.1) and the data rate shaper (VII-A.2).

A.1 Priority packet dropping

We have measured the quality of the priority based packet dropping algorithm by filtering the first 200 frames of the movie with several quality factors (q) ranging from $q = 0.05$ (very low quality) to $q = 0.95$ (high quality). For every resulting video we calculated the average Peak Signal to Noise Ratio (PSNR) in comparison to the WaveVideo encoded video and the average data rate over the 200 frames. These results are shown in Fig. 15. We observe a non-linear correlation between the average data rate and PSNR recognizable. But there is a corelation between q

[2]WaveVideo distinguishes between I- and Delta- frames. All Delta-frames refer to a unique I-frame. A sub-sequence that starts with an I-frame followed by successive Delta-frames is denoted as WGOP. The size of a WGOP is the number of frames in the WGOP, i.e. 1+num(Delta-frames).

and the quality of the signal. For $0.5 \leq q \leq 0.95$, a big reduction of the data rate with a small degradation of quality can be achieved. For example, the PSNR is still good with a value of 36 dB for $q = 0.95$, 31 dB for $q = 0.9$ and 30 dB for $q = 0.8$.

Figure 16 shows four screenshots of our testvideo, filtered to different qualites. The same frame is shown with different quality factors. Nearly no distortion of the picture is visible for $q = 0.9$ and $q = 0.8$ as shown in Fig. 16 (a) and (b). Pictures (c) and (d) show a small, but acceptable degradation of quality for $q = 0.7$ and $q = 0.6$ whilst having a big data rate saving (ref. also to Figure 15).

A.2 Priority data rate shaping

Fig. 17 shows the correlation between q and the target data rate (linear curve), the relation to peak data rate of the video sequence and the average data rate. The filter keeps the output data rate below a certain threshold. Every peak of the datasize of a frame, which would exceed this threshold is cut off.

We use the PSNR as our distortion measure. But for the data rate shaper filter, the PSNR is not well suited. Frames, whose data rate are already below this threshold, are not filtered at all. This means, that they are identical to the original frame, which will result in a PSNR of ∞. Since ∞ values can't be handeled numerically, they are not considered for the calculation of the average PSNR. The number of frames, which are identically to the original are shown in Figure 18. Instead, it would be necessary to use QoS policy-based measurements, with a subjective evaluation for lossy as well as lossless cases. For future measurements of our filter, we will use different measurements techniques.

Of course, to disregard identical images will influence the calculation of the average. That is the reason for the discontinuous PSNR curve in Figure 19. Nevertheless this figure shows the correlation of the data rate and PSNR. Note, that the average data rate has a more linear trend, than shown in Fig. 15 from the measurements of the priority packet dropper and therefore it can be used as a very simple version of a data rate shaper filter, which can also be implemented efficiently in routers. A more elaborated

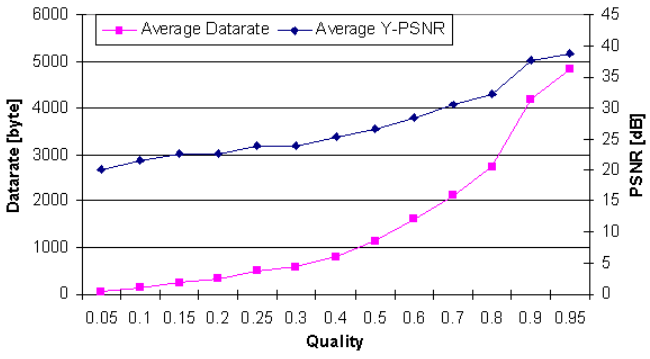

Fig. 15. PSNR and data rate of the priority packet dropping filter.

Fig. 16. Example of the result of the priority based packet dropper with a qualty factor of (a) q=0.9 (b) q=0.8 (c) q=0.7 and (d) q=0.6.

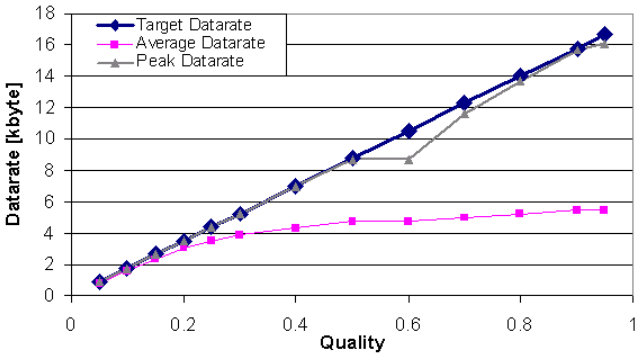

Fig. 17. Target data rate, average data rate and peak data rate of the post filtered video sequence using the data rate shape filter.

but also more complex data rate shaper filter implementation based on semantical information has been presented in [14].

B. Frame Rate Filter Evaluation

We were interested in the bandwidth scalability of the frame rate filters. Therefore, we applied the simple and advanced frame rate filter to the compressed movie and recorded post filter data rate as a function of target frame rate r_{dest}.

As can be seen from Fig. 20, the advanced frame rate filter smoothes the average data rate much more than the simple frame rate filter, where discontinuities occur when switching between 20 and 21 fps, 15 and 16 fps and 4 and 5 fps. However, the decrease in post filter data rate is not linear as can be seen by comparing with the *linear decrease* curve. This is the effect of the different frame sizes of I- and Delta- frames and the variation in the I-/Delta frame pattern. Note also, that the data rate curve increases more in the area of 1 fps until 5 fps than in the remainder. The

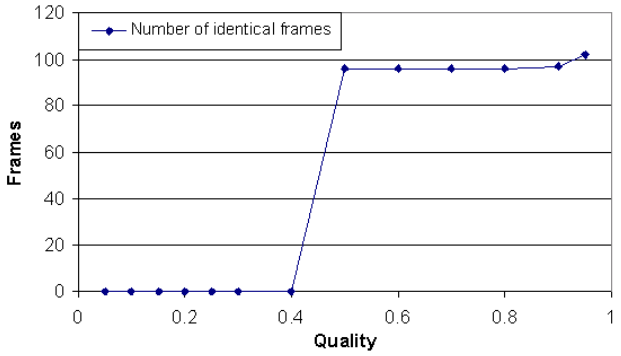

Fig. 18. Number of identical frames produced by data rate shape filter.

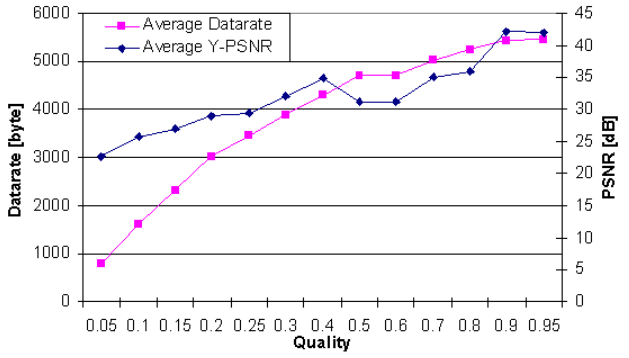

Fig. 19. PSNR and data rate of the data rate shaping filter.

reason for this behaviour is the fact that I-frames are larger than Delta-frames. As the maximum (average) WGOP size was 4.1, a frame rate between 1 and 5 is achieved by only forwarding I-frames. A frame rate of 6 and more requires Delta-frames to be forwarded, which are smaller in size.

VIII. Conclusions

In this paper, we have presented several different filter algorithms for WaveVideo streams; simple syntactical filters as well as a more sophisticated semantical filter that adapts the frame rate of the stream. The syntactical filter uses the RTP header information to make its drop or forward decision; it is efficient and can achieve significant bandwidth reduction while preserving a reasonable media quality. The semantical filter operates on the RTP payload information, in particular the sequence number and the type (I- or Delta-frame) of the packetized WaveVideo frame. This semantical information is beneficial in preserving a good media quality even if many packets need to be dropped, e.g. due to network congestion.

We have integrated our filters in the MASA QoS framework. MASA is a distributed system of cooperating QoS controllers, managers, and brokers. It envolves all entities of the media transmission path, i.e. senders, routers, transcoders, gateways, and receivers. By doing so, MASA achieves end-to-end quality of service through various adaptation meachanism and complementary techniques, such as real time scheduling on the end system. Our WaveVideo codecs are controlled and orchestrated by MASA media managers.

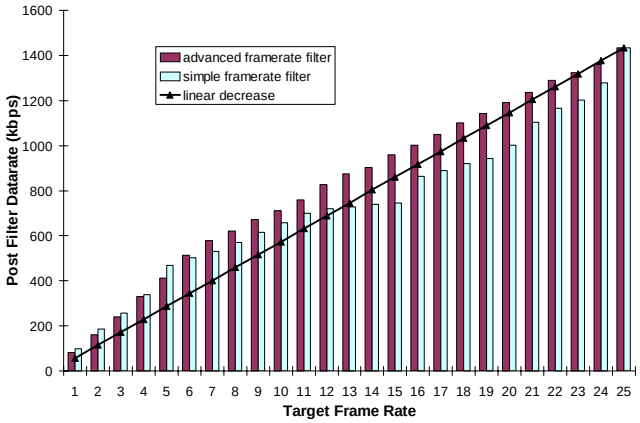

Fig. 20. Comparison of simple and advanced frame rate filter with the linear decrease.

It is worth mentioning that the semantical WaveVideo filter is not restricted to the WaveVideo format but can operate on any arbitrary Wavelet coded video stream format. If this holds also true for the syntactical filter requires further study. However, we are positive that this generalization can be done through a few minor code changes, if any.

Although we are already using Java and the Java Media Framework JMF for a large part of the code, the WaveVideo codec is still native C++ code. The time being, a port to Java seems impractical for performance reasons. However, increasing CPU power, upcoming Java OSs, or optimized JVM (Java Virtual Machine) implementations might change this in the future. This would provide us with the opportunity to run an entirely portable QoS framework on virtually arbitrary platforms.

IX. Acknowledgement

This work has been partly performed in the framework of the MASA project, which is partly funded by Siemens AG, Munich. The author would like to acknowledge the contributions of his colleagues namely Christoph Niedermeier, Michael Krautgärtner, Hannes Hartenstein, Darren Carlson, Linggang Chen, Andreas Schorr, Jochen Wöhrle and Alexander Neubeck.

References

[1] H. Hartenstein, A. Kassler, M. Krautgärtner, A. Schrader, "High Quality Mobile Communication," in *Proceedings of the KIVS'2001 Conference*, 2001.

[2] E. Amir and S. McCanne and M. Verterli, "A layered DCT coder for Internet Video," in *Proceedings of the IEEE Conference on Image Processing ICIP'96*, September 2001.

[3] F. Raspall and C. Kuhmünch and A. Banchs and F. Pelizza and S. Sallent, "Study of packet dropping policies on layered video," in *Submitted to Workshop on Packet Video 2001*, 2001.

[4] G. Fankhauser, M. Dasen, N. Weiler, B. Plattner, and B. Stiller, "WaveVideo - An Integrated Approach to Adaptive Wireless Video," *ACM Monet, Special Issue on Adaptive Mobile Networking and Computing*, vol. 4, no. 4, 1999.

[5] Ralph Keller, Sumi Choi, Dan Decasper, Marcel Dasen, George Fankhauser and Bernhard Plattner, "An Active Router Architecture for Multicast Video Distribution," in *Infocom 2000*, 2000.
[6] Steven MacCanne and Van Jacobson, "Receiver-driven layered multicast," in *Proceedings of ACM SIGCOMM'96*, August 2001.
[7] A. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *RFC2475: An Architecture for Differentiated Services*, IETF.
[8] D. Tennenhouse, J. Smith, W. Sincoskie, D. Wetherall, and G. Minden, "A Survey of Active Network Research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, January 1997.
[9] N. Tripathi, J. Reed, and H. VanLandingham, "Handoff in cellular systems," *IEEE Personal Communications*, Dec. 1998.
[10] C. Perkins, *IP Mobility Support*, October 1996.
[11] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," IETF RFC 1880, Proposed Standard; ftp://ftp.isi.edu/in-notes/rfc1880.txt, January 1996.
[12] Sun, *The Java Media Framework Version 2.0 API*, http://java.sun.com/products/java-media/jmf.
[13] A. Kassler, A. Neubeck, and P. Schulthess, "Classification and Evaluation of Filters for Wavelet Coded Videostreams," *Signal Processing: Image Communication*, vol. 16, no. 8, pp. 795–807, May 2001.
[14] A. Kassler, A. Neubeck, and P. Schulthess, "Real-time Filtering of Wavelet Coded Videostreams for Meeting QoS Constraints and User Priorities," *Proc. of Packet Video Workshop*, Mai 2000.
[15] A. Kassler, A. Neubeck, and P. Schulthess, "Filtering Wavelet based Video Streams for Wireless Interworking," *Proc. of the ICME*, July 2000.
[16] A. Kassler and P. Schulthess, "An End-to-End Quality of Service Management Architecture for Wireless ATM networks," *Proc. of the HICSS 32*, January 1999.
[17] A. Kassler and A. Neubeck, "Self Learning Video Filters for Wavelet Coded Videostreams," *Proceedings of the Int. Conf. of Image Processing (ICIP2000)*, September 2000.

Christian Kücherer studied computer science at the University of Applied Science in Mannheim, Germany, where he focused on communication and distributed systems, object oriented programming and image processing. In 2001 he graduated with a diploma thesis on Quality of Service with adaptive video. Since 2001 he works as a Software Engineer for the Muse platform in the Internet Services Group at NEC Europe Ltd. Heidelberg.

Dr. Andreas Schrader studied electrical engineering at the University of Siegen, Germany, where he focused on information processing and computer graphics. In 1991 he graduated with a diploma thesis on image compression algorithms.
From 1992 to 1998 he was Scientific Employee at the Department of Programming Languages in Siegen. In his PhD thesis (1998) he developed evolutionary algorithms for color quantization and image compression.
From 1998 to 1999 he joined the Computer & Communications Research Laboratories NEC Europe Ltd., Heidelberg, as a Research Staff Member. During that time he was responsible for the development of a service management platform for Internet telephony (MUSE). Since 1999 he works as a Senior Research Staff Member in the Internet Services Group and is responsible for the development of a Quality-of-Service architecture (MASA) in co-operation with industrial and academic partners, the management of UMTS field trials with european telecom providers, as well as for the supervision of master and PhD students. Dr. Schrader is a member of the ACM and has published several scientific papers at international conferences.

Andreas Kassler studied mathematics and computer science at University of Augsburg, Germany, where he focused on communication and distributed systems and image processing. In 1995 he graduated with a diploma thesis on fractal image compression algorithms.
Since 1995 he is with the Distributed Systems Department at University of Ulm, where he continues research in the area of multimedia, wireless networks and Quality of Service Management.
As a project manager he was responsible for the architecture and development of the multimedia subsystem for the WAND (Wireless ATM Network Demonstrator) project. Since 1999, he is involved in several research projects (IST-BRAIN, IST-MIND and MASA), where he is responsible for the design and development of Quality-of-Service architectures. Andreas Kassler is a member of the IEEE and has published several scientific papers at international conferences and journals.

Dr. Oliver Haase received a Diploma in Computer Science from Karlsruhe University in 1993. Thereafter, he joined the group for Practical Computer Science at Siegen University, where he earned a doctorate in 1997. From 1998 to 2000, he was working with the Computer & Communication Research Labs of NEC Europe in Heidelberg; in 1999, he was promoted to Senior Researcher. In Jan 2001, he joined Bell Labs Research in Holmdel, NJ, where he is a member of the Networking Techniques Research Department. Oliver has published numerous papers in international conferences and journals, as well as a textbook about distributed applications (Sockets, Java RMI, CORBA, Jini technology). He has been teaching classes for distributed applications, as well as for databases as a guest lecturer.