

Objective End-to-End QoS Gain from Packet Prioritization and Layering in MPEG-2 Streaming

Daniel Forsgren, Ulf Jennehag, Patrik Österberg

Mid-Sweden University

SE-851 70 Sundsvall

Sweden

{daniel.forsgren, ulf.jennehag, patrik.osterberg}@mh.se

Abstract

Layered video coding as well as prioritized packet scheduling are two well-known methods that may improve the quality of service level in real-time applications with high bandwidth requirements, and are used over packet switched networks. However, it is often difficult to get an idea of, and to quantify, the actual gains that may be achievable, especially from an end-to-end perspective.

In this paper, we present some experimental results from using temporally layered MPEG-2 video combined with basic per-layer IP packet prioritization. The goal has been to find out if a basic scheme is useful at all in combination with this particular source coding method, and if so, how much the objective video quality can be increased during bandwidth-constrained periods. The quality is measured in terms of PSNR and the results are compared to the case of equal packet priority. Also, different packet sizes as well as packet queuing disciplines are used.

We conclude that using even a relatively simple temporal layering strategy in combination with packet prioritization can quite significantly improve the end-to-end quality of MPEG-2 video, especially in moderately bandwidth constrained situations. Furthermore, packet size and queuing discipline is found to have an impact.

Keywords: MPEG-2, layered video, quality of service, traffic shaping

1 Introduction

Real-time video transmission over packet switched networks, such as the IP-based Internet, still remains a challenge after years of research. Realistic transmission of high-quality video streams generally requires high compression ratios, which lead to an increased sensitivity to transmission errors (e.g. packet loss) and make this case especially challenging.

Successful approaches to eliminate or reduce network degradation of real-time media streams include addressing the problem at, or below, the network (IP) level by using physical links with quality of service (QoS) support, enhanced QoS-aware routing protocols [1], support for integrated services/differentiated services etc. In practice, many of these approaches require upgrades of network nodes of a kind that network operators typically are reluctant to introduce, for reasons of complexity or economy. Today, most IP networks remain best effort, although support for quality of service functionality will most likely grow in proportion with the interest for using high-quality media applications over the networks.

The lack of network support for QoS could be partially dealt with using lesser means, by making the best effort of the network even better for some applications or traffic types. This type of solution tries to reduce the impact of any congestion problems closer to network leafs, without relying (at least not exclusively) on QoS functionality in the network as a whole to take care of traffic prioritization. Traffic shaping [2] is often implemented in individual hosts for shaping their outgoing traffic, but it can also be performed on interfaces further into the network.

For example, a router that forwards traffic from a subnet in which many real-time video streams are known to originate, ending up in the aggregated traffic flow on its interface(s), could retain instantaneous information about the currently available bandwidth in each direction. Based on this information (and perhaps even information about throughput along a whole network path) the router could adjust the packet prioritization behavior on each interface according to prevailing local and/or path conditions. The more routers along a communication path that would use this strategy, the better the bandwidth utilization would be. With such a scheme, if a router approaches its current bandwidth limit on a certain interface, it can for example, elect to drop more packets belonging to non-real time applications.

However, in a router where there are lots of aggregated high-bandwidth video data flows, there is a risk that most of the data would end up belonging to the prioritized group, and the prioritization would become meaningless. For the prioritization to be successful in these situations, the router should be able to make more fine-grained prioritization decisions than what is typically feasible. Preferably it should be able to distinguish between, and make prioritizations of, different types of data within a video stream (e.g. several traffic classes per application). This level of granularity is not feasible to implement throughout a large network, but in routers close to leaf networks where the aggregated traffic level is moderate, it can be.

One way of achieving this granularity is to employ a layered source coding method. This would enable nearby network nodes to prioritize certain data inside each video stream. Using layer (i.e. packet) priority information, as well as information about local and perhaps even remote bandwidth availability, routers can queue and drop packets in a “controlled” manner before congestion occurs.

In this paper, we present the results of experiments made using temporally layered source coding in combination with an intermediate router node, which uses class-based queuing to give different priority to packets depending on which video layer their payload belongs to. The goal of the experiments was to see how much the overall end-to-end video quality, in terms of luminance peak signal to noise ratio (PSNR) [3], could be improved in moderately bandwidth-constrained situations using simple variants of such an approach.

2 Layered Video Coding and Transmission Method

Scalable and layered video transmission schemes have been widely studied before [4, 5]. However, their actual use in combination with different fine grained packet prioritization schemes and the improvement in perceived end-to-end video quality that can be achieved, has not received as much coverage. This chapter introduces the video coding format, layering method and transmission methods that we have used throughout our experiments.

The video-coding format used is the widely adopted MPEG-2 [6]. We have used multiple pre-encoded MPEG-2 streams, each with a constant bit rate (CBR) of

8 Mbps, consisting of the popular reference sequences “BBC3”, “Cactus and comb”, “Flower garden”, “Mobile and calendar”, “Susi”, and “Table tennis”, all of which are available from the Tektronix FTP site [7]. The pre-encoded streams are divided into video layers offline, while the transmission and reception of the layered video streams take place in real-time.

The temporal layering method we have used assigns the different picture types in a group of pictures (GOP) to a certain layer depending on the importance of the picture type. Even though the GOP structure is not mandatory in MPEG-2, the need of a repeating pattern is required in order to apply a temporal layer method. Therefore the GOP structure of twelve pictures per GOP, of which three are P pictures, is assumed. Figure 1 shows the GOP structure and how it has been mapped to the video layers. The importance of the layers is in numerical order where layer zero is most important.

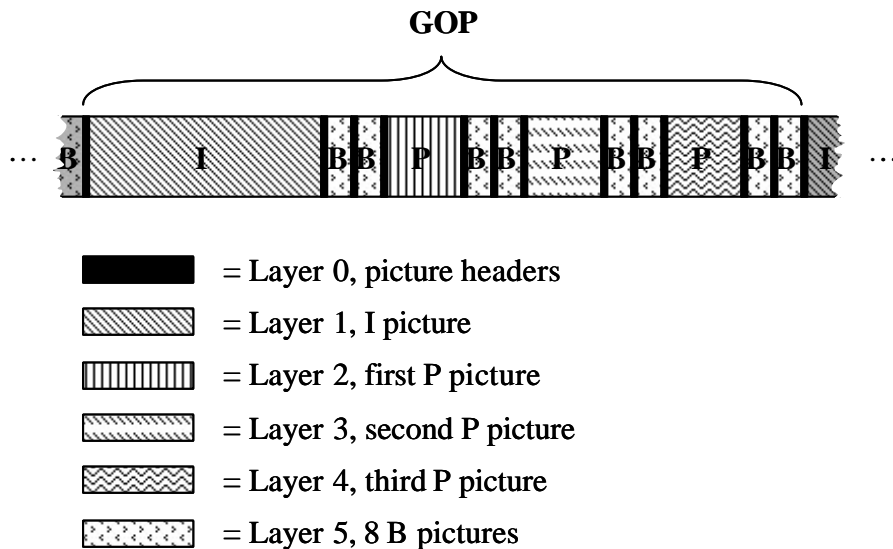


Figure 1: The GOP structure and its mapping to video layers

More accurate approaches exist for determining the packet priority. It can be calculated from the macroblock (MB) encoding type, the motion vectors (MV) of each MB, total packet size and picture-level header occurrences [8]. The MVs are relevant since their length and direction affect the error propagation [9]. Another parameter that may be relevant to consider is the dependence count, the number of times a MB is referenced.

Our experiments have been performed in a setup, laid out as shown in figure 2. The sending application, running in the send host, aims at transmitting each of the different priority classes with CBR. That is, even though in the video stream, there might be a burst of consecutive packets belonging to the same priority class, these packets are distributed evenly among the other packets when transmitted. We consider this per-GOP packet interleaving a good measure, since it does not incur any significant additional delay, and reduces the need for buffering bursts in network queues when there are few parallel video streams. Due to the chosen redistribution of the packets, a real time streaming video receiver will experience a short initial delay, which corresponds to the time it takes to receive all the packets belonging to the first GOP.

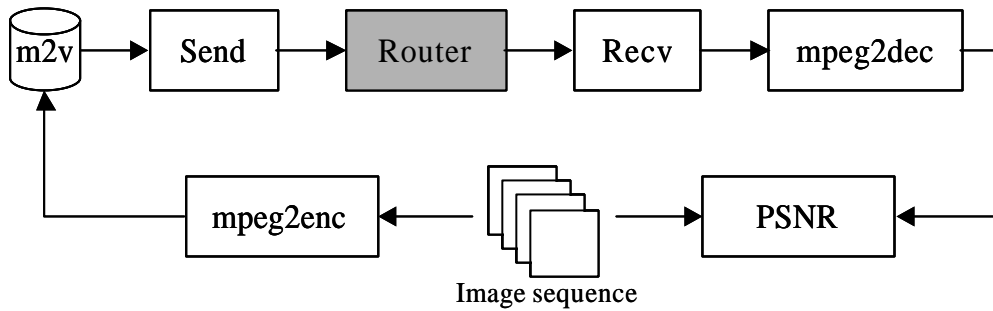


Figure 2: Experimental setup

For the receiver to be able to process packets arriving in this non-deterministic order, the sender adds a sequence number (SN) at the beginning of each packet, thus adding 4 bytes of overhead. The sender transmits video data using UDP/IP. Therefore, besides the payload of video data of each packet, the packets will consist of 8 bytes of UDP header and another 20 bytes of IP header. The layout of a video packet can be seen in figure 3.

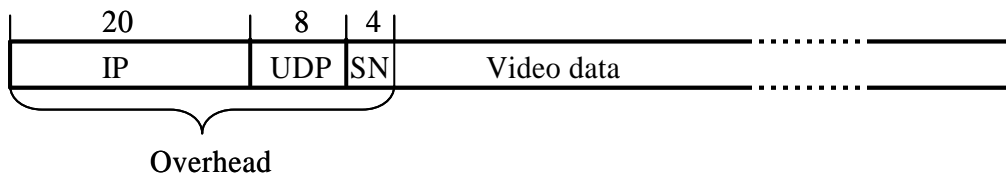


Figure 3: Video packet layout

When all the video data packets that make up a clip have been transmitted, the sender transmits an end of transmission (EOT) packet, which is always delivered to the receiver. This is only for making the experiments, where we send thousands of clips in rapid succession, easier to track, and would not necessarily be needed in real world applications.

As the receiver receives a packet, it extracts the SN and copies the video data in the packet payload to the corresponding position of an assigned memory area. When an EOT packet is received, the receiver copies the data from the memory area to a file. The PSNR of the video in the file is later calculated off-line using the original image sequence as a reference. The decoder used in the experiments is the MSSG MPEG-2 video codec [10]. The decoder has been slightly modified to be able to decode erroneous MVs and illegal variable length codes (VLC), functionality to handle loss of picture headers has also been included.

3 Network Setup and Parameters

As described in section 2, a sender host has acted as a video server, and transmitted video clips separated into temporal layers, encapsulated in UDP/IP packets. On their way to the recipient, the packets have been routed through an intermediate router node, in which the outgoing interface has been configured for class-based traffic control. The network interfaces involved in the experiments were all of fast Ethernet type, with a maximum transmittable unit (MTU) of 1500 bytes. The router node has been using Linux 2.4 kernel traffic filtering and control functionality to map incoming packets to the proper traffic class on the outgoing network interface, and

queued accordingly. The mapping has been based on IP header information, e.g. the type of service (TOS) value.

For the division of outgoing traffic into traffic classes, the router has been using hierarchical token bucket (HTB) class based queuing, which is a more efficient and clean alternative to the standard class based queuing (CBQ) [11] that is a component of the Linux 2.4 kernel family [12]. The packet queuing disciplines we have chosen to use and compare, in both the prioritized and the non-prioritized cases, are simple first in first out (FIFO) and stochastic fair queuing (SFQ). FIFO requires little introduction, SFQ was discussed in [13] and is basically a lightweight fair queuing discipline that uses collected packet statistics to shape the traffic.

FIFO and SFQ are two of the most popular disciplines available, and they are also suitable for use in our proposed scenario (in leaf routers with limited aggregated bandwidth). Furthermore, neither one of these disciplines are controlled by a large number of parameters, which makes them good candidates for objective comparison. In contrast, many of the more complex queuing disciplines, such as random early detection (RED) (which is primarily designed for use in backbone routers) have a large number of parameters, which are typically chosen and tuned using approximate rules-of-thumb to obtain the desired behavior. In our experiments, however, we have kept all queuing parameters fixed for all experiments.

In the network, the variables in the experiments have been the total available bandwidth, whether packets are prioritized or not, and which queuing strategy is used. The network had no notion of packet size or statistics of the particular video clip currently being transmitted.

In the non-prioritized cases, we have simply used a single traffic class (set for either FIFO or SFQ queuing), into which all data packets have been filtered, irrespective of which layer the data happens to belong to. The total available bandwidth in the single class has ranged between 5000 and 8800 kbps in the experiments, with less spacing between data points in the more interesting areas of the bandwidth interval. Since we have used a video compression scheme that is very sensitive to data loss, we have judged this area to reside in the upper quarter of the bandwidth interval.

In the prioritized cases, we have used a prioritization scheme where packets are filtered into a layer-specific traffic class, depending on the type of data the packet is carrying as indicated by the IP header. The total bandwidth range available to all the classes was the same as in the non-prioritized case. To achieve an efficient, yet simple and practically feasible, prioritization between the traffic classes, we let the classes carrying information from high-priority video layers get a guarantee of having a greater percentage of their expected (average) bandwidth available when needed. Our chosen values for guaranteed bandwidth are shown in table 1.

Layer	Content	Average bw	Guaranteed bw
0	Header data	280kbps	100%
1	I-frames	2100kbps	90%
2	P-frames 1	900kbps	70%
3	P-frames 2	900kbps	50%
4	P-frames 3	900kbps	30%
5	B-frames	3000kbps	20%

Table 1: Layer prioritization

Each class has been allowed to use more bandwidth when available. However, no bandwidth has been considered excessive before all classes have obtained required bandwidth up to their respective guaranteed levels. No class is allowed to use bandwidth exceeding a certain peak level, which has equaled the total available bandwidth in each experiment. In order to get a consistent queuing behavior, the sum of the guaranteed bandwidths for all classes has been ensured to stay well below the minimum available total bandwidth (i.e. the bandwidth of the main parent class) in all experiments.

The average bandwidth figures for the video layers have been calculated over all the video clips and packet sizes used. We have found these figures, and especially the relation between them, to be representative enough for all the scenarios, thus we have kept them constant for all experiments.

The output was in the form of a number of received video clips, which were finally decoded and used to calculate the average PSNR impact of the network conditions. Each experiment has been performed a large number of times and with all of the six video clips [7], in order to improve the accuracy of the results. All experiments have also been repeated for each of the three fixed packet sizes (550, 1000 and 1450 bytes payload).

4 Results

Figure 5 shows an overview of the PSNR performance under each of the four cases. The payload per packet is in this case fixed to 550 bytes. It is evident that the layer prioritization creates an overall lift of the PSNR curves when compared to the non-prioritized case with the same packet queuing strategy. Figure 5 also shows that using SFQ improves the video quality when compared to FIFO, especially in the non-prioritized case.

Figure 6 shows the corresponding results with the only difference being that a payload of 1000 bytes has been used in each video packet. It can be clearly seen that increasing the packet size has a positive impact on the PSNR performance in all bandwidth-constrained situations, and the prioritized case is still superior. The increase in PSNR from the use SFQ instead of FIFO is approximately doubled.

Finally, in figure 7 the corresponding plots for the largest packet size used, 1450 bytes payload, can be found. As expected, the PSNR performance is best in this case. SFQ has an even greater influence than in the smaller packet cases, probably because it was configured for an MTU of 1500 bytes.

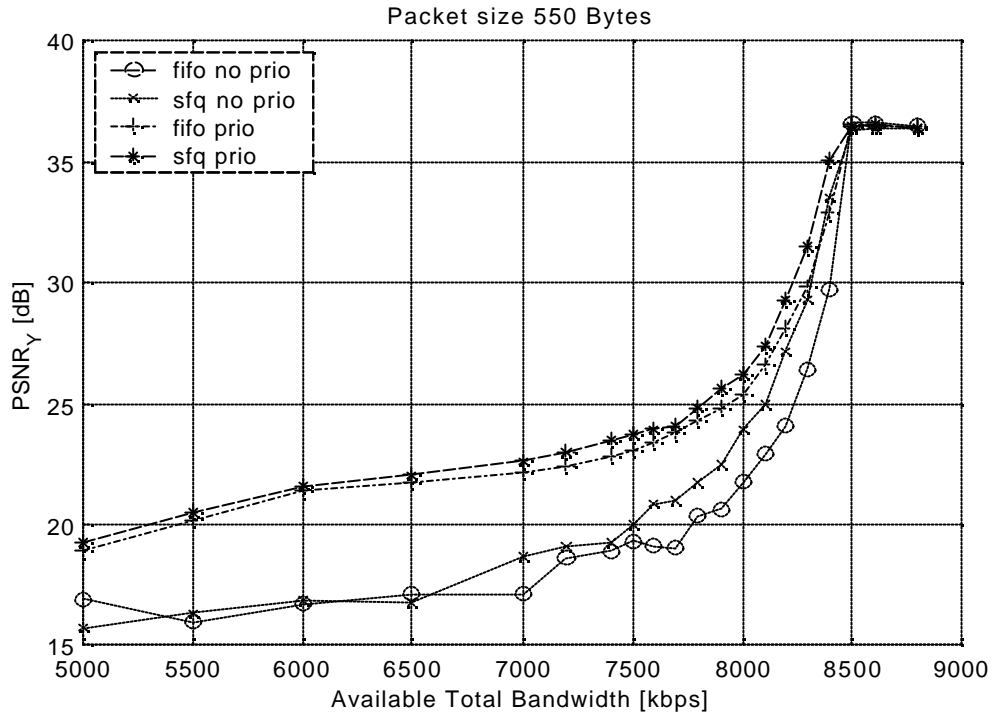


Figure 5: Performance of each scheme with 550 bytes payload

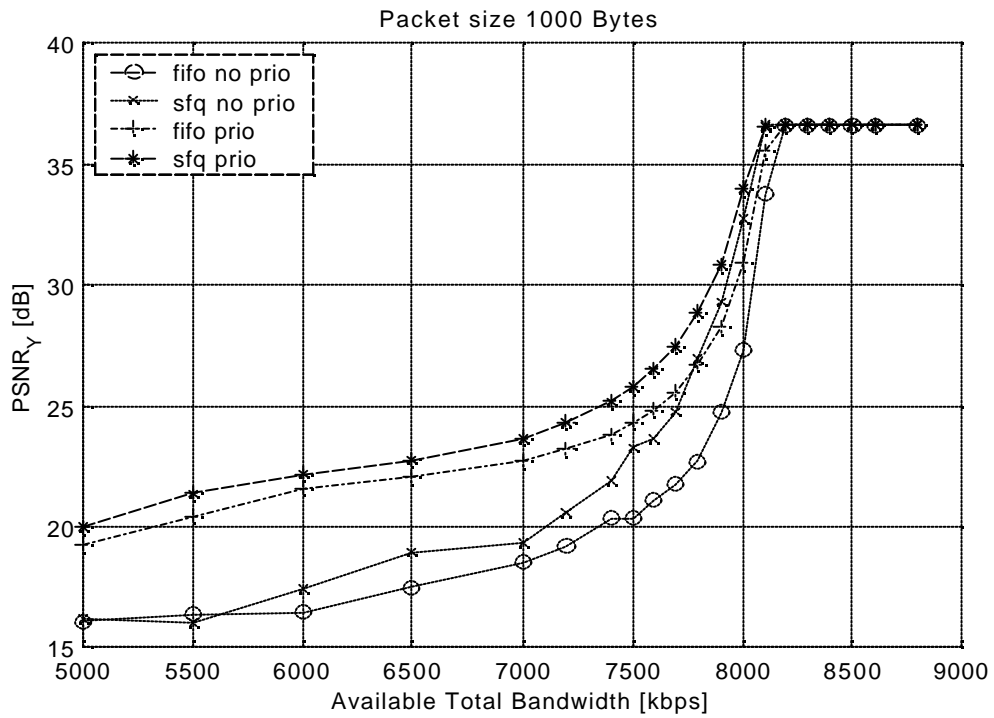


Figure 6: Performance of each scheme with 1000 bytes payload

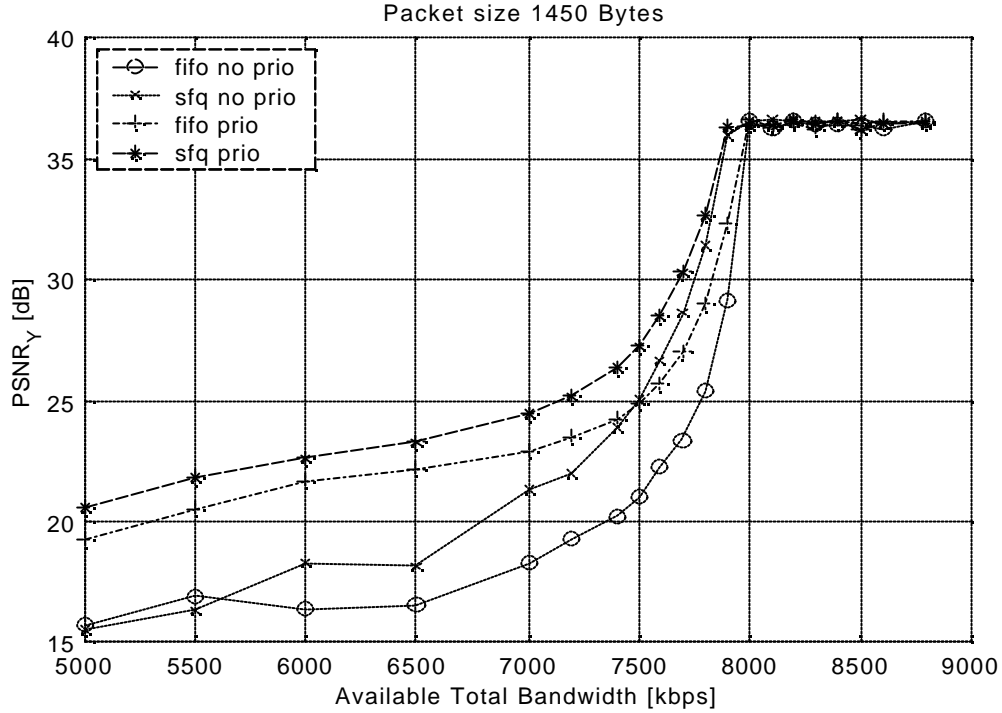


Figure 7: Performance of each scheme with 1450 bytes payload

5 Conclusions and Future Work

The results of these experiments show an increase in objective video quality, in terms of PSNR, when packet prioritization is applied. At lower available bit rates, 1 Mbps or more below the bit rate of the sender, this increase is about 3-5 dB. At bit rates closer to the bit rate of the sender, the increase is about 2-3 dB.

We can also see that the queuing strategy affects the end-to-end video quality; SFQ outperforms FIFO. At bit rates closer than 500 kbps to the bit rate from the sender, this influence is greater than that of the prioritization.

Furthermore, it is evident that packet size also has an impact on the objective video quality, and should be chosen as large as possible. An obvious reason for this is that a smaller packet size leads to extra overhead, but the difference is too big to be solely explained by the extra overhead. Another reason is likely the fact that with an error sensitive source coding schemes like MPEG-2, it is due to error propagation better to lose large chunks of data at a lower rate than small chunks at a higher rate.

Future work could include to make similar experiments using other source coding schemes (e.g. MPEG-4) together with more advanced and fine-grained layering methods (e.g. spatial scalability coupled with temporal scalability), and a greater variety of network settings, in order to further increase the overall end-to-end PSNR efficiency. Optimally, the PSNR curve should get a more linear behavior over the entire bandwidth spectrum. This is dependent on all tiers in the layered video transmission system, but the layered source-coding scheme is probably the most important.

It would also be interesting to make experiments where multiple layered video streams are sent over the same data prioritizing link, and to add competing

background traffic to the aggregated traffic flow, to test the scalability of the approach in terms of PSNR.

References

- [1] Chen, S. and Nahrstedt, K., "An overview of quality-of-service routing for the next generation high-speed networks: problems and solutions", IEEE Network Magazine, Vol. 12, No. 6, November-December 1998
- [2] Bechler, M., Ritter, H., Schafer, G. and Schiller, J. "Traffic shaping in end systems attached to QoS-supporting networks" Proceedings of the Sixth IEEE Symposium on Computers and Communications, 2001
- [3] ANSI T1.801.03-1996, "Digital transport of one-way video signals-parameters for objective performance assessment", American National Standards Institute, 1996
- [4] Zhang, T. and Xu, Y., "Unequal packet loss protection for layered video transmission", IEEE Transactions on Broadcasting, June 1999
- [4] Amir, E., McCanne, S., and Vetterli, M., "A layered DCT coder for Internet video", Proceedings of the IEEE International Conference on Image Processing, Lausanne, Switzerland, September 1996
- [6] ISO/IEC 13818-2, "Information technology - Generic coding of moving pictures and associated audio information: Video", 1996
- [7] Tektronix FTP site, <ftp://ftp.tek.com/tv/test/streams/Element/MPEG-Video/625/>, July 31, 1999
- [8] Jitae, S., JongWon, K. and Kuo, C.-C. J., "Relative priority based QoS interaction between video applications and differentiated service networks", Proceedings of International Conference on Image Processing, Vancouver, September 2000
- [9] de los Reyes G., Reibman, A., and Chang, S., "A corruption model for motion compensated vide subject to bit errors" Proceedings of Packet Video Workshop, New York City, May 1999
- [10] MPEG Software Simulation Group (MSSG), "MPEG-2 Encoder / Decoder, Version 1.2", <ftp://ftp.mpeg.org/pub/mpeg/mssg/>, July 19, 1996
- [11] Devera, M. and Cohen, D., "HTB Linux queuing discipline manual", <http://luxik.cdi.cz/~devik/qos/htb/>, December 19, 2001
- [12] Hubert, B. et al, "Linux Advanced Routing & Traffic Control HOWTO", <http://www.linuxdoc.org/HOWTO/Adv-Routing-HOWTO.html>, December 6, 2001
- [13] McKenney, P., "Stochastic Fairness Queuing", Proceedings of INFOCOM, San Fransisco, June 1990