Darmstadt University of Technology

Retransmission Scheduling in Layered Video Caches

Michael Zink, Jens Schmitt, Ralf Steinmetz

05 March 2001

Technical Report TR-KOM-2001-07

Industrial Process and System Communications (KOM)

Department of Electrical Engineering & Information Technology
Merckstraße 25 • D-64283 Darmstadt • Germany

Phone: +49 6151 166150
Fax: +49 6151 166152
Email: info@KOM.tu-darmstadt.de
URL: http://www.kom.e-technik.tu-darmstadt.de/

Transport Issues for Wide-area Distribution Infrastuctures based on Patching

Michael Zink[1], *Jens Schmitt*[1], *Ralf Steinemetz*[2]

1
KOM, Dept. of Electrical Engineering & Information Technology
Darmstadt University of Technology
Merckstr. 25 • D-64283 Darmstadt • Germany

2
German National Research Center for Information Technology,
GMD IPSI
Darmstadt, Germany

Michael.Zink@kom.tu-darmstadt.de, Jens.Schmitt@kom.tu-darmstadt.de, Ralf.Steinmetz@tu-darmstadt.de

Abstract

In contrast to classical assumptions in Video on Demand (VoD) research, the main requirements for VoD in the Internet are adaptiveness, support of heterogeneity, and last not least high scalability. Hierarchically layered video encoding is particularly well suited to deal with adaptiveness and heterogeneity support for video streaming. A distributed caching architecture is key to a scalable VoD solution in the Internet. Thus, the combination of caching and layered video streaming is promising for an Internet VoD system, yet, requires thoughts about some new issues and challenges. In this paper, we investigate one particular of these issues: how to deal with retransmissions of missing segments for a cached layered video in order to meet users' demands to watch high quality video with relatively little quality variations. We devise a suite of fairly simple retransmission scheduling algorithms and compare these against existing ones by simulative experiments.

Keywords

Internet VoD, Caching, Layered Video

1 Introduction

1.1 Motivation

In the last few years, the Internet has experienced an increasing amount of traffic stemming from the use of multimedia applications which use audio and video streaming [1]. This increase will continue and even be reinforced since access technologies like ADSL and cable modems enable residential users to receive high bandwidth multimedia streams. One specific application which will be enabled by future access technologies is Video on Demand (VoD). VoD allows clients to watch a large variety of video content via the Internet, from small video clips up to movies. One type of VoD is True VoD (TVoD) [2] which allows users to watch a certain video at any desired point in time and, in addition, offers the same functionality as a standard VCR (fast forward, rewind, pause, stop). The challenges of providing TVoD in the Internet are manifold and require the orchestration of different technologies. Some of these technologies like video encoding are fairly well understood and established. Other technologies like the distribution and caching of video content and the adaptation of streaming mechanisms to the current network situation and user preferences are still under investigation.

Existing work on TVoD has shown caches to be extremely important with respect to *scalability*, from network as well as from video servers' perspective [3]. Scalability, of course, is a premier issue if a TVoD system is considered to be used in the global *Internet*. Yet, simply reusing concepts from normal Internet Web caching is not sufficient to suit the special needs of video content since, e.g., popularity life cycles can be very different [4].

Besides scalability, it is very important for an *Internet* TVoD system to take into account the "social" rules implied by TCP's cooperative resource management model, i.e., to be *adaptive* in the face of an (incipient) network congestion. Therefore, the streaming mechanisms of an Internet TVoD system need to incorporate end-to-end congestion control to prevent unfairness against TCP-based traffic and increase the overall utilization of the network. Note that traditionally video streaming mechanisms rely on open-loop control mechanisms, i.e., on explicit reservation and allocation of resources. As it is debatable whether such mechanisms will ever be used in the global Internet, e.g., in the form of RSVP/IntServ [5], we do not assume these but build upon the current best-effort service model of the Internet which is based on closed-loop control exerted by TCP-like congestion control. Yet, since video transmissions need to be paced at their “natural” rate adaptiveness can only be integrated into streaming mechanisms in the form of quality degradation and not by “shifting” traffic in the time domain as for elastic traffic like, e.g., FTP transfers. An elegant way of introducing adaptiveness into streaming is to use layered video encodings [6] as it allows to drop segments (the transfer units) of the video in a controlled way without the high computational effort of, e.g., adaptive encodings as described in [7].

However, while the combination of caching and adaptive streaming promises a scalable and “Internet-conform” TVoD system it also creates new challenges for the design of such a system. One particular issue is that video content can only be cached in the form as it has been transmitted, i.e., it potentially consists of successive “steps” of different quality levels corresponding to the different layers. For subsequent requests for that video it must thus be decided if segments from missing layers are retransmitted and if so which ones. The scheduling of these retransmissions affects the perceived quality of the cached video content in a significant way since it is very important that quality variations are minimized as they are disturbing for users [8]. Therefore, we focus in this paper on how to schedule retransmissions in order to minimize quality variations for users that are served from the video cache.

1.2 Outline

After this motivation, we briefly want to introduce the basic components of our overall approach towards scalable adaptive video streaming in the Internet, before, in Section 3, we discuss some related work.

In Section 4, we then focus on the particular problem of how to schedule retransmissions of segments from missing layers of a video towards a cache. We briefly demonstrate that this is a complex problem, which is why we devise a number of heuristic algorithms. In Section 5, these heuristics are compared among each other and against existing approaches by simulations as well as we investigate their dependency on parameters like, e.g., the number of layers in which a video is encoded.

In Section 6, we summarize our findings, draw some conclusions and give a brief outlook to future work.

2 Scalable Adaptive Streaming (SAS)

2.1 Scalability - Caching

As with traditional web caches, caches for TVoD systems allow to store content closer to users, reduce server and network load and increase the system’s fault tolerance. Yet, in contrast to web caches the characteristics of the data to be stored is very different. High quality video files are much larger than most web pages and therefore different caching strategies are used in caches for VoD systems. Furthermore, the distribution process for video files is complicated by the fact that the transmission is much more time and bandwidth consuming.

Let us briefly describe our video caching architecture. As caching method we employ so-called write-through caching[*]. With write-through caching a requested stream is either forwarded “through” the proxy cache or it is streamed via multicast and clients and proxy caches join this multicast group if the cache replacement strategy decides to store the requested video on the proxy cache. Subsequent clients can then be served from the proxy cache (see Figure 1). This technique reduces the overall network load

*. Adopted terminology from memory hierarchies.

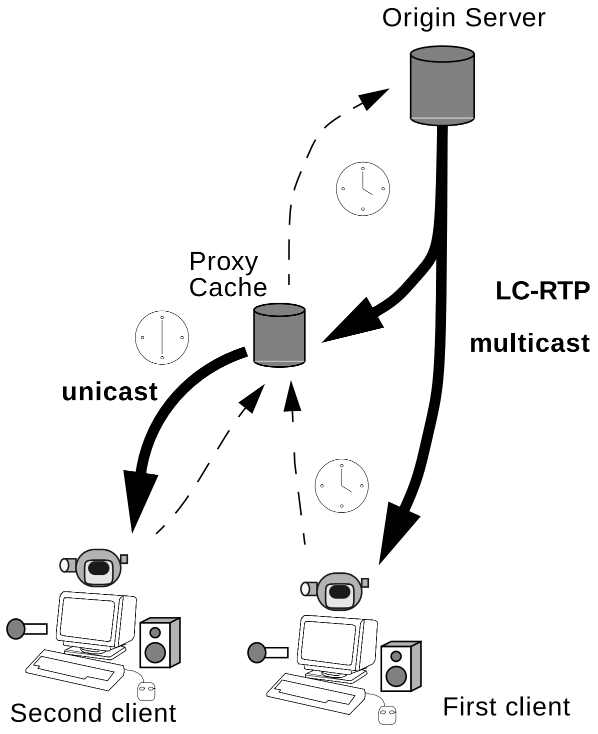

Figure 1. Video distribution.

in a TVoD system compared to a method where the video is transported to the cache in a separate stream using a reliable transmission protocol (e.g., TCP) [9]. On the other hand, write-through caching requires a reliable multicast protocol to recover from packet losses. In [10], we present the design and implementation of such a protocol which fits particularly well in a TVoD architecture.

2.2 Adaptiveness - Layered Video

To enable congestion control for streaming applications results in quality adaptation in contrast to elastic applications that can be spread over time. However, this quality adaptation does not solely serve congestion control purposes but also satisfies the needs of the large variety of heterogeneous clients that exist in the Internet. Layered video, i.e., video that is encoded in base and enhancement layers which have hierarchical relationships, represents a suitable method to allow for this quality adaptation although there are other alternatives like adaptive encoding or switching between different encodings. Yet, the latter are less attractive for caching purposes since they do not possess the subset relationship of layered encoding and thus lead to transmissions which are difficult to cache. Figure 2 illustrates how a layered video might be stored on a cache after its initial (congestion controlled) transmission.

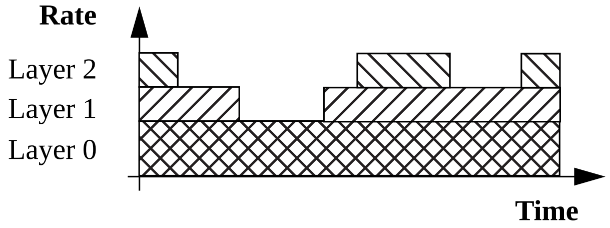

Figure 2. Initial cached video quality.

Obviously, the cached copy of the video exhibits a potentially large number of missing segments from different layers. The exact “shape” of a cached video content is a function of the congestion control mechanism being used. There have been several proposals how to achieve TCP-friendly congestion control using layered video transmissions, e.g., [11] or [12]. We do not focus on this particular aspect of

SAS here, but build on these proposals since they are already very efficient. Our emphasis is on an orthogonal issue: the scheduling of retransmissions of missing segments of a cached video.

2.3 Retransmission Scheduling

With SAS, it is very likely that videos are not cached in their best quality when they are cached for the first time. However, for subsequent requests which shall be served from the proxy cache it may be unattractive to suffer from the possibly very bad or strongly varying quality experienced by the initial transmission of the video that has been selected for caching. Therefore, missing segments of the cached video should be retransmitted to enable higher quality service from the proxy cache to its clients. The most interesting issue here is how to schedule the retransmissions, i.e., in which order to retransmit missing segments, in order to achieve certain quality goals for the cached video content. A further design issue for retransmission scheduling is when to do it.

2.3.1 Retransmission Time

There are two basic alternatives when to do retransmissions:

- Directly after the initial streaming process: the cache starts requesting missing segments without waiting for further requests for a certain video.
- During subsequent requests: the proxy cache serves subsequent requests but, simultaneously, also orders missing segments from the origin server.

While the first alternative ensures that a cached video’s quality is improved as fast as possible, the second alternative inherits the advantage of write-through caching that any bandwidth between proxy cache and origin server is used only if a client request is directly related to it. This is a major advantage in environments where bandwidth between origin server and proxy cache is scarce. Assuming that the proxy cache is located fairly close to the clients and that origin server and proxy cache are rather distant from each other, and furthermore taking into account that we assume an Internet TVoD system we selected the second alternative for our investigations. Note that a further argument for the second alternative is that the quality of the cached contents is a function of request patterns.

2.3.2 Scheduling Goals

The rationale for making an effort to schedule retransmissions in an intelligent way is that the presentation quality for users that are served from the proxy cache can be enhanced. Therefore, we need to make explicit what constitutes a quality enhancement, i.e., we need a goal for retransmission scheduling algorithms to strive for. First of all, it is obvious that any retransmission of missing segments increases the average quality of a cached video. Therefore, all algorithms we investigate use as much bandwidth as available between origin server and proxy cache to retransmit missing segments. That means with respect to average quality they are all the same. However, it is commonly assumed that users react very sensitive to quality variations of a video [8]. Hence, a retransmission scheduling algorithm that tries to avoid or even decrease quality variations for a cached video can be considered superior to others which do not take this into account. The negative effect of quality variations has two dimensions:

- the frequency of variations, and
- the amplitude of variations.

Hence, the goal of retransmission scheduling should be to minimize, both, the frequency and amplitude of quality variations.

To state the scheduling goal more formally, let us define some terms:

h_t - number of layers in time slot t, $t = 1,..., T$

z_t - indication of a step in time slot t, $z_t \in \{0,1\}$, $t = 1,..., T$

Here, we assume without loss of generality a slotted time with slots corresponding to the transmission time of a single (fixed-size) segment and that all layers are of the same size. We can now introduce what we called the *spectrum* of a cached layered video *v*:

(3)

$$s(v) = \sum_{t=1}^{T} z_t \left(h_t - \sum_{j=1}^{T} z_j h_j \right)^2$$

(4)The spectrum captures the frequency as well as the amplitude of quality variations. The amplitude is captured by the differences between quality levels and average quality levels where larger amplitudes are given higher weight due to squaring these differences. The frequency of variations is captured by the z_t. Only those differences are taken into account that correspond to a step in the cached layered video. While the spectrum as defined in (3) looks very similar to the usual variance of quality levels for the cached video, it is important to note that the introduction of the z_t takes into account the frequency of changes of the quality levels.

The retransmission scheduling goal for a video *v* can now be stated as the minimization of the spectrum $s(v)$.

5 Related Work

[13] were among the first that investigated cache replacement algorithms for multimedia streams. Yet, their work did not take into account transport issues and layered encoded video.

[3] put very much emphasis on the examination of a scalable transport infrastructure for cache replacement algorithms. However, the inclusion of adaptiveness as a requirement for Internet VoD was not yet considered.

In [14], the authors propose an interesting scheme of caching only the beginning of video streams. While this allows to decrease the setup latency for clients and to accommodate variable bit rate transmission channels it does not address the scalability and adaptiveness issues.

Like us, [15] considers the combination of caching and layered video, yet, the latter only for the support of heterogeneous clients but not for congestion control purposes. Furthermore, the emphasis of their work is on optimal cache replacement decisions viewed over *all* videos stored in a cache. We, however, assume a two-stage decision process where in the first stage a video is selected for storage in a cache and then the retransmissions of missing segments are scheduled independent from the cache status of other videos. While this represents a restricted problem it ensures that the overall problem still remains manageable.

Really close to our work and actually inspiring for our work was [16]. However, we extend their work by focussing on the development and comparison of different retransmission scheduling algorithms which are more flexible and performing better than the one presented in [16].

6 Algorithms for Retransmission Scheduling

In this section, we first discuss the problem complexity of retransmission scheduling by looking at optimal retransmission scheduling. Since computation of optimal retransmission schedules is computationally infeasible or at least intensive, we then present some heuristic schemes. One of them has been proposed in [16], whereas the others are devised by us based on shortcomings of the former.

6.1 Optimal Retransmission Scheduling

The goal of retransmission scheduling is to minimize the spectrum of an already cached layered video subject to the constraint that any available bandwidth is used for retransmissions. This constraint ensures that a cached video is further enhanced even if for all time slots the same quality level is reached, i.e., the spectrum would be 0.

A formulation of optimal retransmission scheduling as mathematical program is given in Figure 3.

d_t	-	number of retransmitted layers for time slot *t*
v'	-	the cached video after retransmissions
H	-	the maximum number of layers
$\tilde{B}(\bar{t})$	-	estimated amount of overall retransmission capacity for all time slots till $\bar{t}$

Minimize $s(v')$ (7)

subject to (8)

$$\sum_{= 1}^{t} d_t = \tilde{B}(\bar{t} \qquad /\bar{t} = 1, \ldots, 7 \qquad (9)$$

$$+ d_t - h_{t-1} - d_{t-1} \leq H \qquad /t = 1, \ldots, 7 \qquad (10)$$

$$_{-1} + d_{t-1} - h_t + d_t \leq H \qquad /t = 1, \ldots, 7 \qquad (11)$$

$$I - h_t \geq d_t \geq (\qquad /t = 1, \ldots, 7 \qquad (12)$$

Figure 3. Optimal retransmission scheduling model.

Here, the overall available retransmission capacity is modeled as an estimate. Yet, in our investigations we assume a constantly available bandwidth, i.e.,

$$3(\bar{t}) = \frac{B}{T} \times ,$$

where B is the overall retransmission capacity for the video. This is certainly a simplifying assumption, yet, our algorithms do not depend on it.

We observe that optimal retransmission scheduling is a discrete non-linear optimization problem. As such it is - to the best of our knowledge - analytically intractable. It is very similar to the quadratic assignment problem, which is known to be NP-complete [17]. To illustrate the complexity of retransmission scheduling let us also consider the search space for an exhaustive search: if we assume that in each time slot at least one layer is missing then we obtain as a *lower* bound for the size of the search space: $\binom{T}{B}$, e.g., for 100 time slots and a retransmission capacity of 50 this amounts to 1.534×10^{93} possible ways of reordering missing segments (and this is only a very loose lower bound). Thus, an exhaustive search for reasonable values of the number of time slots *T* is computationally infeasible.

12.1 Heuristics for Retransmission Scheduling

12.1.1 Window-Based Lowest Layer First (W-LLF)

The first heuristic we want to look at has been proposed in [16]. It is fairly simple and we call it Window-Based Lowest Layer First (*W*-LLF), because the proxy cache always looks a certain number of time slots ahead of the current playout time and requests retransmissions of missing segments from the server in ascending order of their layer levels. To ensure that the retransmitted segments do not arrive after their playout time (t_p) to the current client, a prefetching offset O_p for the examined time window is introduced. O_p should be chosen sufficiently large such that O_p > RTT for the transmission path between server and cache at all times. Overall, the time window $[t_p + O_p, t_p + O_p + W]$ slides over the video in discrete steps of length W until it is finished (t_{end}). The operation of the algorithm is further illustrated in Figure 4.

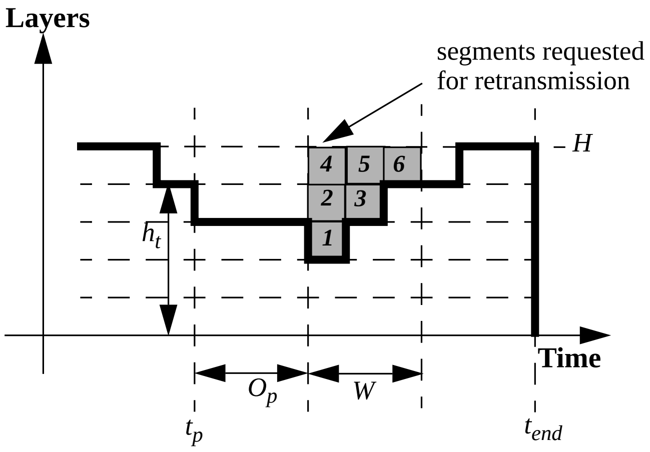

Figure 4. *W*-LLF operation.

W-LLF bears some obvious disadvantages:

- If, e.g., an already complete area (all layers are entirely cached) is scanned, no retransmissions are scheduled for this prefetching window, although there might very well be later parts of the video which could benefit from retransmissions.
- It may be possible that the currently available bandwidth between origin server and proxy cache would allow the transmission of more segments than those that are missing in the current prefetching window and again additional segments could be requested from the server to allow for a faster quality upgrade of the cached video.

Although, these obvious drawbacks might be eliminated by extensions of the *W*-LLF algorithm, they exhibit a fundamental weakness of *W*-LLF: the restriction of scheduling missing segments for retransmission only for a certain number of time slots ahead. Therefore, *W*-LLF is likely to be rather short-sighted with respect to the scheduling goal of minimizing the spectrum of videos cached on the proxy. In the following, we introduce a new kind of retransmission scheduling algorithms that eliminates this restricted view.

12.1.2 Unrestricted Priority-Based Heuristics

The problems with *W*-LLF as described above lead us to the idea to avoid the use of a prefetching window for retransmission scheduling. That means we take an unrestricted look at all missing segments ahead of the current playout time (plus the prefetching offset O_p) when making requests for retransmissions from the origin server. Note that our algorithms still send periodic retransmission requests to the server (every W time slots) to ensure on the one hand that retransmissions and playout to the client are kept synchronized and on the other hand that the modified shape of the cached video due to retransmitted segments can be taken into account by the scheduling algorithms.

Furthermore, we want to introduce more flexibility into the scheduling decisions by the notion of general priorities for retransmission scheduling decisions instead of rigidly always choosing the segments with the lowest layer level.

In the following, we describe three heuristics of the more general class of unrestricted priority-based retransmission scheduling algorithms.

Unrestricted Lowest Layer First (U-LLF)

This algorithm is very similar to *W*-LLF because it uses as priority solely the layer level. In contrast to *W*-LLF, however, it always scans the interval $[t_p + O_p, t_{end}]$ in order to request missing segments from the server (every *W* time slots).

Unrestricted Shortest Gap Lowest Layer First (U-SG-LLF)

Considering the definition of the spectrum in Section 2.3.2 and taking into account our scheduling goal of minimizing the spectrum, we can observe that there are, in principle, two ways to decrease the spectrum of a video: to increase the lowest quality levels (which is taken care of by choosing the lowest layer levels first) *and* to close gaps in the video, i.e., reduce the number of $z_t \neq 0$. The latter is not captured by simply using layer levels as priorities. Figure 5 gives an illustrative example how the spectrum is affected by the closing of gaps.

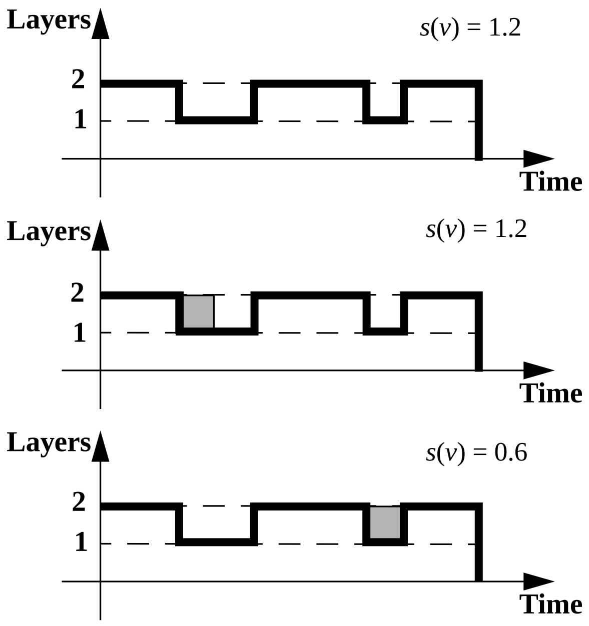

Figure 5. Influence of closing gaps on spectrum.

The influence of closing gaps on the spectrum can potentially be quite high. Therefore, in contrast to *W*-LLF and U-LLF, we now use a prioritization of the missing segments which also takes closing of gaps into account. We do so by simply sorting the segments according to the length of the gap they belong to and then sort these further by their layer levels. The resulting heuristic we called Unrestricted Shortest Gap Lowest Layer First (U-SG-LLF).

Unrestricted Lowest Layer Shortest Gap First (U-LL-SGF)

Since it is by no means clear, which sorting criterion, i.e., gap length or layer level, should go first we also tried the variant where missing segments are first sorted by their layer level and then sorted further by gap lengths, which we called Unrestricted Lowest Layer Shortest Gap First (U-LL-SGF).

13 Simulations

To compare the different retransmission scheduling algorithms from the previous section and to investigate their dependency upon different parameters, we performed a number of experiments based on a simulation environment implemented (in C++) particularly for that purpose.

The simulations are performed in the following manner: For each simulation an instance of a layered video on the proxy cache is randomly generated. Here, we modeled such a layered video instance as a simple finite birth-death process since it is the result of the congestion-controlled video transmission which restricts state transitions to direct neighbor states. $\{0,..., H\}$ is the state space and birth and death rate are chosen equal as $l - 1/\sqrt{3}$ (for all states) which results in a mean length of 3 time units for periods with stable quality level [†]. We use a discrete simulation time where one unit of time corresponds to the transmission time of a single segment. In Figure 6, an example video instance generated in this way is given.

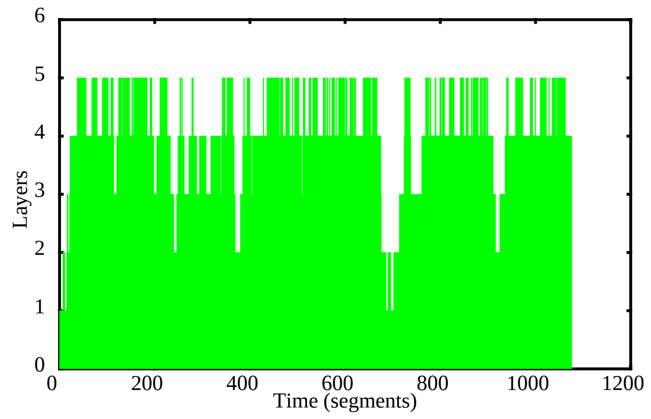

Figure 6. Randomly generated layered video on the cache.

Our simulation environment allows then to apply the different algorithms described in Section 4 and to vary parameters like, e.g., the bandwidth that is available for retransmissions between origin server and proxy cache. During the simulations, the spectrum (as defined in Section 2.3.2) of the cached video instances is continuously calculated to compare the different algorithms and their performance depending on parameters as, e.g., the available bandwidth. In all simulations we assumed a prefetching offset *of* $O_p = 5$ segments.

13.1 Comparison of the Heuristics

At first, we performed a series of 1000 simulations with all retransmission scheduling algorithms from Section 4 where all parameters were chosen identical (except the windows sizes for *W*-LLF). This large sample ensured that the 95% confidence interval lengths for the spectrum values were less than 0.5% of the absolute spectrum values for all heuristics. The results for the evolution of the spectrum values for the different algorithms are shown in Figure 7.

These results indicate that there is a significant gain with respect to the spectrum of the cached video for the unrestricted retransmission algorithms proposed by us in comparison to *W*-LLF. Of course, if window sizes are chosen large enough for *W*-LLF it improves and finally approaches U-LLF.

Among the unrestricted algorithms there seems to be little difference such that one may argue for the use of the simplest algorithm, i.e., U-LLF.

†. We have to admit that the parameter choice is rather arbitrary. However, simulations with other values showed no significant impact on our results.

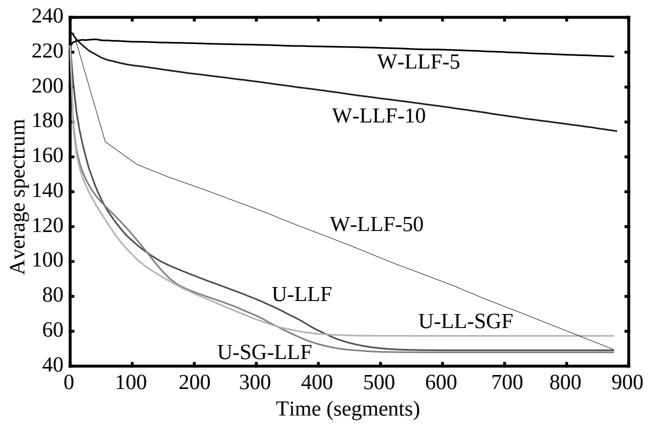

Figure 7. Average spectrum of 1000 simulation runs for each heuristic. (10 layers, retransmission bandwidth = 2, window size = 5 (*W*-LLF-5), window size = 10 (*W*-LLF-10), window size = 50 (*W*-LLF-50)).

13.2 Parameter Dependency Analysis

In the following, we investigated the heuristics' dependencies on certain parameters. For all of these simulations, we only used the U-SG-LLF heuristic since it showed the best performance of all heuristics in the experiment of the preceding section.

13.2.1 Number of Layers

For this simulation, we varied the number of layers per cached video from 5, 10 to 20 layers. To isolate the effect of encoding videos with different number of layers, the available retransmission bandwidth was scaled in proportion to the amount of layers, i.e., for 5 layers we assumed 2 segments of retransmission bandwidth per time unit, for 10 layers we used 4, and for 20 layers 8. For each of these 3 alternatives we ran 1000 simulations and calculated again the average of the spectra over time.

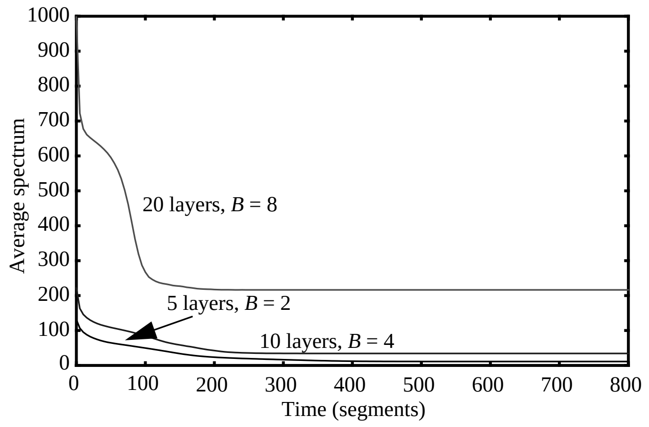

Figure 8. Different number of layers.

As Figure 8 shows, the spectrum converges for each of the three alternatives. Yet, the higher the number of layers the higher the average spectrum. This is intuitive because the more fine-grained the layered encoding the more variations may be introduced during a congestion-controlled transmission and the harder it is for the retransmission scheduling to smooth these variations.

13.2.2 Available Retransmission Bandwidth

In the next set of simulations, the effect of different amounts of available retransmission bandwidth on the performance of U-SG-LLF was investigated.

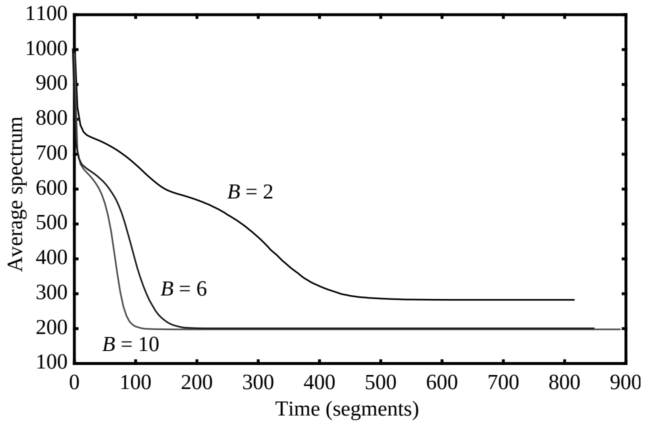

Figure 9. Different amounts of available retransmission bandwidth.

Not surprisingly, the spectrum converges faster with a higher available retransmission bandwidth. The reason for the very similar spectrum curves for B = 6 and 10 is due to sufficient retransmission bandwidth for both cases which allows to retransmit all missing parts of the rear 3/4 of the cached video. Due to the prefetching offset, missing segments from the beginning cannot be retransmitted and therefore a spectrum of 0 is not achieved.

13.2.3 Initial Transmission Quality

Finally, we performed a series of simulations where different initial transmission qualities were assumed resulting in cached videos where the maximum number of cached layers is lower than the maximum number of layers for the original video. In contrast to the preceding experiments, we did not sample the spectrum values but used a single simulation since the striking effects can be shown in more detail. For each simulation, the ratio between the maximum of cached layers (MCL) and the maximum of original layers (MOL) was modified.

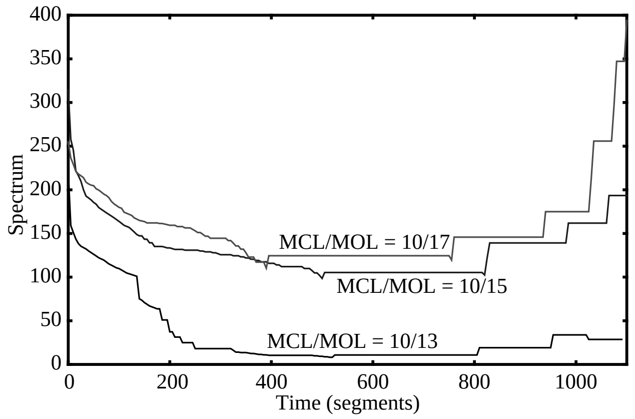

Figure 10. Influence of initial transmission quality.

As Figure 10 shows, spectrum values start to rise again for the last third of the video. This effect is especially pronounced for the worse initial transmission qualities.

Looking at the cached video that results from the retransmission scheduling heuristic (U-SG-LLF) in Figure 11 sheds light on the reason for this effect. We observe that the retransmission scheduling "built a staircase" at the end of the cached video which, of course, is not good with respect to the minimization of the spectrum. The reason for this behavior is that the algorithm only considers missing segments ahead of the playout time ($t_p + O_p$). Thus, if all gaps are closed the algorithm starts to request segments from the next layer starting from $t_p + O_p$. This happens several times leading to the staircase shape exhibited in Figure 11.

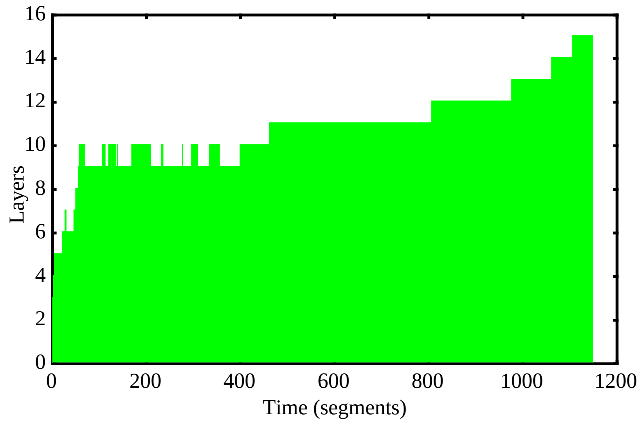

Figure 11. Cached video after retransmission phase.

13.3 Totally Unrestricted Heuristics

As a consequence of the results from Section 5.2.3, we wanted to see how the modification of the heuristics in a way that retransmissions are not limited to segments that are located after $t_p + O_p$ could cure the problem the heuristics had with bad initial transmission qualities. However, it has to be observed that such a totally unrestricted retransmission scheduling algorithm bears the possibility that retransmitted segments may arrive too late for the current client and might thus be retransmitted in vain if no other client ever requests that video. We are thus loosing some of the attractiveness of write-through caching. On the other hand, it also offers the chance to obtain a complete copy of the video on the cache.

We repeated the same simulations from Section 5.2.3 with the now totally unrestricted version of U-SG-LLF. The results are shown in Figure 12.

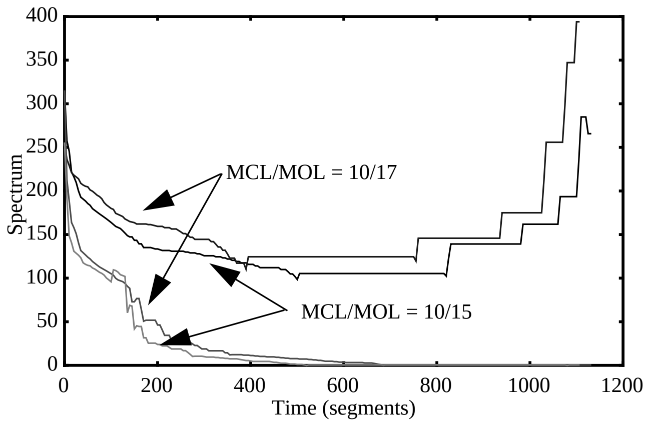

Figure 12. Influence of initial transmission quality for totally unrestricted heuristics.

Obviously, the problem of rising spectrum values is solved. This observation is reinforced when we compare the cached video as it results from the totally unrestricted heuristic in Figure 13 to its counterpart in Figure 11.

However, in order to assess how many segments would be scheduled for retransmission which could not be viewed any more by the current client, we also recorded these “late” segments: with MCL/MOL = 10/17 55% and with MCL/MOL = 10/17 56% of the retransmitted segments arrived too late. This is certainly a substantial amount of late segments and thus one has to make a decision here between generating a fairly smooth cached video and using all available retransmission bandwidth to benefit the current client.

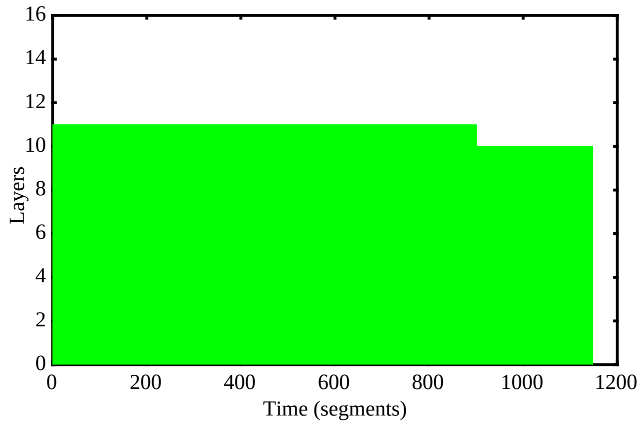

Figure 13. Cached video after retransmission phase for totally unrestricted heuristic.

14 Conclusions and future work

Recent work has shown that layered encoded video is a technique that supports adaptive streaming well. High scalability for VoD in the Internet can be achieved by a distributed caching architecture. Our SAS (Scalable Adaptive Streaming) approach combines both caching and adaptive streaming and promises a scalable “Internet-conform” TVoD system. Our work, in this paper, is focused on the problem how to deal with retransmissions of missing segments for a cached layered video in order to meet users’ demands to watch high quality video with relatively little quality variations. Inspired by [16], we developed and compared different retransmission scheduling algorithms from the general class of unrestricted priority-based heuristics to tackle this problem. Our simulation results indicate that this class has the potential to improve existing algorithms significantly. In addition, we investigated the dependency of the algorithms on system parameters. These showed that the heuristics could not cope well with poor initial transmission qualities. Therefore, the concept of totally unrestricted heuristics was presented and shown to cure the problem albeit at the cost of a considerable number of late segments.

Overall, the insights gathered from our simulative experiments encourage us to integrate adaptive streaming by layered video encodings into our VoD system[‡] as part of our future work. This will allow us to verify our simulative results for the retransmission scheduling algorithms in “real-life” scenarios. Furthermore, from the algorithmic perspective, we are going to investigate how the decision to cache a certain video and how to schedule its retransmissions can be integrated with each other, respectively, how they affect each other.

‡. See http://www.kom.tu-darmstadt.de/kom-player.

15 References

[1] S. McCreary and kc claffy. Trends in Wide Area IP Traffic Patterns, May 2000. http://www.caida.org/outreach/papers/AIX0005/.

[2] T. D. Little and D. Venkatesh. Prospects for Interactive Video-on-Demand. *IEEE Multimedia*, 1(3):14–25, May 1994.

[3] C. Griwodz. *Wide-area True Video-on-Demand by a Decentralized Cache-based Distribution Infrastructure*. PhD thesis, Darmstadt University of Technology, Darmstadt, Germany, April 2000.

[4] C. Griwodz, M. Bär, and L. C. Wolf. Long-term Movie Popularity in Video-on-Demand Systems. In *Proceedings of ACM Multimedia'97*, pages 340–357, November 1997.

[5] R. Braden, D. Clark, and S. Shenker. RFC 1633 - Integrated Services in the Internet Architecture: an Overview. Informational RFC, June 1994.

[6] J.-Y. Lee, T.-H. Kim, and S.-J. Ko. Motion Prediction Based on Temporal Layering for Layered Video Coding. In *Proceedings ITC-CSCC'98*, pages 245–248, July 1998.

[7] K. Shen and E. J. Delp. Wavelet Based Rate Scalable Video Compresssion. *IEEE Transactions on Circuits and Systems for Video Technology*, 9(1):109–122, February 1999.

[8] R. Gopalakrishnan, J. Griffioen, G. Hjalmtysson, C. Sreenan, and S. Wen. A Simple Loss Differentiation Approach to Layered Multicast. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies 2000, Tel-Aviv, Israel*, pages 461–469, March 2000.

[9] R. Frederick, J. Geagan, M. Kellner, and A. Periyannan. Caching Support in RTSP/RTP Servers. Internet Draft, March 2000. Work in Progress.

[10] M. Zink, C. Griwodz, A. Jonas, and R. Steinmetz. LC-RTP (Loss Collection RTP): Reliability for Video Caching in the Internet. In *Proceedings of the Seventh International Conference on Parallel and Distributed Systems: Workshops*, pages 281–286, July 2000.

[11] R. Rejaie, M. Handley, and D. Estrin. RAP: An End-to-End Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies 1999, New York, NY, USA*, pages 395–399, March 1999.

[12] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings of the ACM SIGCOMM '00 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication 2000, Stockholm, Sweden*, pages 43–56, August 2000.

[13] R. Tewari, H. Vin, A. Dan, and D. Sitaram. Resource-Based Caching For Web Servers. In *Proceedings of SPIE/ACM Conference on Multimedia Computing and Networking (MMCN), San Jose, USA*, January 1998.

[14] S. Sen, J. Rexford, and D. Towsley. Proxy Prefix Caxching for Multimedia Streams. In *Proceedings of the Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies 1999, New York, NY, USA*, pages 1310 – 1319, March 1999.

[15] J. Kangasharju, F. Hartanto, M. Reisslein, and K. W. Ross. Distributing Layered Encoded Video through Caches. To appear in *Proceedings of the Tenth Annual Joint Conference of the IEEE Computer and Communications Societies 2001, Anchorage, NY, USA*, April 2001.

[16] R. Rejaie, H. Yu, M. Handley, and D. Estrin. Multimedia Proxy Caching for Quality Adaptive Streaming Applications in the Internet. In *Proceedings of the Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies 2000, Tel-Aviv, Israel*, pages 980 –989, March 2000.

[17] M. Garey and D. Johnson. *Computers and Intractability.* W. H. Freeman and Company, San Francisco, California, USA, 1979.