ITC-SS21/Copyright 2010 IEICE

Supporting Scalable Video Codecs in a P2P Video-on-Demand Streaming System

Simon Oechsner, Thomas Zinner, Jochen Prokopetz, Tobias Hoßfeld
University of Würzburg, Institute of Computer Science, Germany
Email: {oechsner|zinner|prokopetz|hossfeld}@informatik.uni-wuerzburg.de

***Abstract*—There are currently two complementing and concurrent trends in the Internet. The first is the rise of video streaming and especially Video-on-Demand (VoD) as the most popular application for end users, expressed by a continuously increasing amount of video traffic. In this context, P2P is seen as a promising technology for providing video content efficiently to the users while reducing the cost for the content providers. The second trend is the diversification of end user devices used for watching these videos. This is reflected by the Scalable Video Codec (SVC) extension to the state-of-the-art H.264 video codec, which allows for a stream to be separated into substreams of varying quality and size. In this paper, we propose and evaluate a P2P VoD architecture based on the Tribler application which is enhanced to support such a SVC video. Thus, the proposed system is able to adapt the video quality on-the-fly to the network situation and access capabilities of the user devices.**

I. Introduction

In recent studies on the traffic distribution in the Internet, e.g., [1], video streaming and especially Video-on-Demand (VoD) such as offered by YouTube are shown to gain more and more popularity. The traffic share of this application class grows accordingly. However, since videos in high quality consume a large amount of upload capacity for streaming, the resources needed to provide a video service to many users are costly. This is especially true when a client-server architecture is used to distribute the content. This problem is shared with traditional content distribution, i.e., file download. Here, an incorporation of the end users into the distribution effort can be achieved by the use of Peer-to-Peer (P2P) technology. Consequently, P2P is a prime candidate also for efficiently streaming video content. Clients like Tribler [2] already support VoD services based on well-known mechanisms from traditional file-sharing.

In parallel to this development, the number and variety of end user devices that can play back a video in acceptable quality has grown. A video may be streamed to a mobile device connected via UMTS, a laptop in a WLAN or to a LCD TV with a broadband connection to the Internet. Current hardware developments like mobile Internet devices with specialized processors to enable a fast and efficient decoding of HD videos foster this trend [3]. However, due to their different capabilities and access bandwidths, there is no single ideal video stream for all devices. A mobile device typically has a smaller resolution and a smaller download capacity, and thus would need a video stream in a lower quality and with lower bandwidth demands in comparison to a HDTV connected via DSL or even FTTH.

To be able to support heterogeneous end user devices, a content provider might offer the same video in different levels of quality, resulting in one video stream per version. A more efficient solution is the recent addition of Scalable Video Codecs (SVC) [4] to the popular H.264 coding standard [5]. This codec allows for a separation of a single source video file into layers which can theoretically be extracted from the stream with no additional coding effort. Thus, a device with low access bandwidth can only download the base layer that is necessary for a playback of the video, while a device with a good connectivity might also request enhancement layers for a better quality of the video.

While approaches and implementations for P2P VoD with Multi Description Coding (MDC) as well as SVC live streaming exist, to our knowledge there is currently no architecture combining SVC with P2P VoD. Therefore, we propose and evaluate in this paper a P2P VoD system based on Tribler that supports a SVC video. We consider one of the scalability dimensions defined in the SVC standard, namely temporal scalability, and focus on the adaptivity of the mechanism to different network conditions. To this end, we conduct a simulative performance evaluation and observe the effect of different network load situations on the efficiency of the system.

The rest of the paper is structured as follows. We will give a short introduction into the background of P2P VoD systems as well as the SVC standard and review related work in Section II. In Section III, we will describe the modifications we made to the Tribler architecture in order to support a SVC video stream. The model used for the simulative performance evaluation is covered in Section IV, while the results of that evaluation are presented in Section V. Finally, we conclude the paper in Section VI.

II. Background and Related Work

We will first shortly give an overview on the two major topics we consider in this paper, namely the H.264/SVC video codec extension and P2P VoD streaming. Then, we will review other work done in this field.

A. Background

H.264/SVC The video codec H.264/SVC [6] [4] is based on H.264/AVC, a video codec used widely in the Internet, for instance by video platforms (e.g., YouTube, GoogleVideo) or video streaming applications (e.g., Zattoo). H.264/AVC is a so called single-layer codec, which means that different video files have to be provided to support different end user devices. The Scalable Video

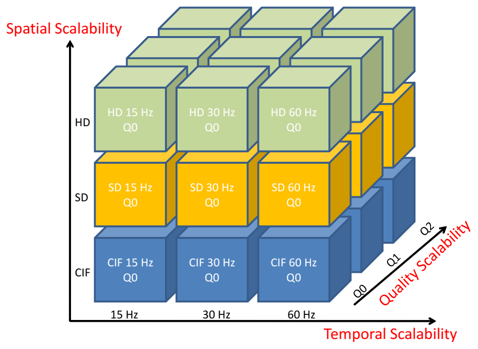

Fig. 1. SVC Cube, illustrating the possible scalability dimensions for a video file

Coding (SVC) extension of H.264/AVC enables the encoding of a video file at different qualities within the same layered bit stream. This includes besides different resolutions also different frequencies (frames displayed per second) and different qualities w.r.t. Signal-to-Noise Ratio (SNR). These can be considered as a special case of spatial scalability with identical picture size for base and enhancement layers. These three dimensions of enhancements are denoted as spatial, temporal and quality scalability.

Figure 1 gives an example of different possible scalabilities for a video file. The scalable video file can be watched in three different temporal resolutions (15Hz, 30Hz, 60Hz), three different spatial resolutions (CIF, SD, HD) and three different quality resolutions (Q0, Q1, Q2). The left bottom "subcube", CIF resolution with 15 Hz and quality Q0, is the base layer which is necessary to play the video file. Based on this layer different enhancement layers permit a better video experience with a higher resolution, better SNR or higher frame rate, respectively. The more subcubes along any of the three axes are available the higher the quality in this respect is. If all subcubes are available the video can be played back in highest overall quality. If all subcubes within quality Q0 are available, the video can be played back in HD-resolution with 60 HZ, but only with a low SNR quality.

P2P VoD Some work exists on mesh-based P2P overlays utilized for distributing single-layer video content in an on-demand fashion. In [7], a server-based VoD streaming system is assisted by BitTorrent clients to reduce the load on the servers and improve the scalability of the architecture. The authors of [8] considered the playout deadline of chunks in the piece selection, enabling a BitTorrent client to show the video while downloading it. This was enhanced in Tribler [2], which also uses priority sets for the chunks that are based on their playback time. Additionally, Tribler introduced the Give-to-Get (G2G) peer selection strategy in the unchoking process [9], which addressed the problem of the missing reciprocity in data exchange that appears in VoD overlays.

B. Related Work

Since the advent of the SVC extension to the H.264/AVC codec, some work has been conducted to include this in a streaming overlay. A live streaming architecture based on a multicast tree is presented in [10]. It utilizes the substreams of SVC to adapt the content that is forwarded along the tree to the capacities of the recipients. In [11], this is extended to an overlay spanning a long-haul network and featuring user devices with different content demands. A number of other works for scalable P2P streaming exists, which we cannot describe due to space limitations. However, all of these approaches implement a live streaming architecture, i.e., a system where the same content is viewed at roughly the same time by the consumers. In contrast, we consider a VoD system here, where peers watch the video at different points in time. This means that the distribution of the complete file and therefore the chunk selection are of much greater importance than in a live streaming system. Also, many live streaming systems rely on a tree topology, in contrast to the mesh-based topology considered here.

An approach that is more similar to our system utilizes Multi-Description Coding (MDC) techniques to allow for a scaling of the video quality [12]. The authors neglect the redundancy normally introduced by MDC in order to achieve efficient streaming. They argue that due to the dependencies in a layered approach such as SVC the received quality is always lower than in a MDC approach without these dependencies. However, in their evaluation, they neglect the effects of stalling as well as seeding times on the system, which have a large impact on the availability of especially the base layer in our approach. Also, a centrally managed system is assumed, which is not realistic for a large-scale P2P system. Moreover, it is not fully clear how the content is distributed in the simulated overlays, since there is no description of the according mechanisms. In contrast, the system presented here is based on a deployed and therefore realistic application, namely Tribler.

III. Architecture Description

The P2P VoD architecture we evaluate is based on Tribler, which in turn is a video streaming BitTorrent adaptation. We will first give an overview on the key mechanisms of Tribler before describing the modifications we introduced in order to support the streaming of a SVC video.

A. Tribler

The VoD client of Tribler adapts the two most important mechanisms of BitTorrent, which are the peer selection strategy in the unchoking process and the piece or chunk selection strategy. Details on these mechanisms can be found in [13]. Since we adapt the chunk selection strategy for our approach, we will shortly describe its implementation in Tribler.

The main difference between a download functionality as offered by BitTorrent and a VoD service as offered by Tribler is that a user of the latter watches the video while downloading it. Thus, the timing for downloading the parts of the complete video file becomes critical, while chunks can be downloaded in any order in a file-sharing network. In particular, chunks in Tribler need to be downloaded roughly in order so that a continuous playback of the video can be ensured.

To this end, the rarest-first chunk selection of BitTorrent is replaced by a strategy based on priority sets. From

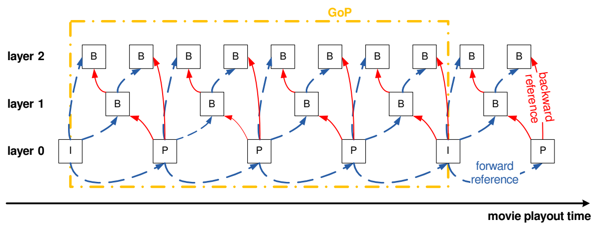

Fig. 2. Frame structure of a video with temporal scalability

the current playback position, all chunks until the end of the movie are separated into three sets. The high-priority set contains all chunks with frames from the playback position until 10 seconds after it, while the mid-priority set contains the following 40 seconds of the movie. The remainder comprises the low-priority set. Chunks are first downloaded from the high-priority set, following an in-order strategy within that set. Afterwards, the chunks in the mid-priority set are downloaded, and finally the chunks of the lowest priority, both according to the BitTorrent rarest-first mechanism.

B. Adaptation for a SVC Video

In order to be able to support a SVC video with temporal scalability, we changed the format of the file that is exchanged in the Tribler swarm, as well as the chunk selection strategy. We will describe these changes in the following.

Format of a Video with Temporal Scalability In this work, we only consider the temporal scalability of a video in SVC. Basically, this means that the frames of the complete video are separated into layers, with each additional layer doubling the frame rate of the video (cf. Figure 2). Since the enhancement layers are referencing all layers below them, they cannot be played out without these layers. Only the base layer contains frames that exclusively reference frames in the same layer, meaning that this layer can be played out by itself.

In this work, we limit our evaluation to a video with 3 layers, the base layer and two enhancement layers. However, the approach and results are applicable to a video with an arbitrary number of layers.

Because the layers are provided via different sub streams, these streams have to be synchronized in order to play the movie back correctly. This can be done, for instance, by appending some bytes indicating the frame number for each frame. This is only necessary for the temporal dimension since different spatial or quality layers can be derived from the according NAL extension header. Thus, an additional amount of a few bytes is sufficient for synchronizing the different sub streams. This overhead is neglected in our simulation study.

Adaptation of the Shared Video File In BT and Tribler, one file is shared per swarm, which is separated into chunks and blocks. In order to be able to discern between different quality layers of the SVC video, we share a number of files in the same swarm, one for each layer. Each of these files, which contains either the base layer or an enhancement layer, comprises its own set of chunks and blocks, just like if it were a single file to be shared.

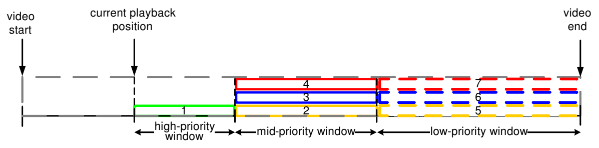

Fig. 3. Adapted chunk selection priority sets

Accordingly, the information in the torrent-file has to be structured to reflect this format and to allow peers to derive the number of chunks to download per layer. Thus, each layer has an ID that together with the chunk ID within a layer allows to identify all chunks of the complete movie file.

Adaptation of the chunk selection Our aim is to download chunks so that a video quality is attained that can be supported by the network capacity. Therefore, we prioritize lower layers over higher ones, while still keeping the set separation and rough in-order download strategy of Tribler (cf. Fig. 3). In the high-priority set, we only download the base layer of the video to make sure we can always play back the video and to avoid stalling times. This means we never download the enhancement layers from the beginning of the video, which is a minor limitation of our architecture and could be circumvented by a longer buffering time. Beginning with the second set, we first download chunks of the base layer according to rarest first. If all of these are already downloaded or no chunk from that layer can be selected, we start downloading chunks from enhancement layer 1, also choosing the rarest chunk first. In general, we only download chunks from a higher enhancement layer if all chunks of the lower layers are locally available or cannot be selected, with one exception that is described below.

Finally, we have the same selection process in the low-priority set, which is only considered if all chunks from the other two sets are downloaded or unavailable from other peers. We also adapted the sizes of the priority sets to 180sec and 360sec for the first and the second priority set, respectively.

During our evaluation, we found that due to the strict prioritization of the base-layer chunks, the content from the enhancement layers was distributed much less, which led to a lower overall quality of the video at the peers. In order to utilize seeders better, we introduce a check whether a local peer is unchoked by a seeder or not. By default, the local peer prefers enhancement layer chunks if it is unchoked by a seeder, if it cannot download a chunk from the high-priority set. It will still download base layer chunks in the mid- and low-priority sets from seeders if no enhancement layer chunks are eligible. We will compare the strategies with and without seeder distinction in Section V.

Adaptation of the playback strategy In Tribler, the video playback is stalled whenever frames are not available in time for their playback. Since we want to exploit the properties of a scalable video file, we adapt this strategy to only stall the video if frames from the base layer are missing. In case frames from enhancement layers are missing, we play out the video with the highest number of layers attainable with the locally existing blocks. As a consequence, a peer that has watched the complete movie does not necessarily have downloaded the complete video file, since it might have missed some chunks from enhancement layers. By default, no chunks are requested that cannot be played out. Thus, a peer might never become a seed. We therefore use the expression serving time instead of seeding time for the interval a peer has finished watching the movie but is still online and providing its completed chunks to the swarm.

Additionally, we buffer the video for 60s before starting to play it back. Since we do not consider a constant bit-rate video (CBR), but dimension the system according to the mean bit rate, we aim at having enough data available so that the varying data rate of the video does not lead to a buffer underrun during playout. On the other hand, a waiting time of one minute should still be tolerable for end users. Still, a more thorough investigation of this parameter is part of our ongoing work.

IV. Simulation Model

In our performance study, we simulate one Tribler swarm that shares a SVC video file with temporal scalability. We use a self-written Java simulator that includes all of the important mechanisms of Tribler and our adaptations of that system. If not stated otherwise, we have two classes of peers with different upload bandwidths. These are based on realistic upload access speeds for DSL1000 and DSL2000 connections, and are set to 128kbps and 192kbps, respectively. We model the uplink of the peers as the only network bottleneck and use a flow-based underlay model to simulate the data transfer of blocks. Thus, the underlay topology apart from the access speeds does not affect the results, since we do not consider additional bottlenecks in the core network. Each data flow, i.e., each block transmission, has an initial delay of 10ms to model the TCP handshake. Concurrent flows share the bottleneck bandwidth fairly.

The shared video is based on an episode of a popular TV show and has a length of 22min. The file has a total size of 55.5MB and a total bit rate of 336kbps. It is separated into three layers using the Joint Multi-View Video Model (JMVM) [14], using a GoP size of 16 embedded frames to achieve temporal scalability. Table I gives a detailed overview on the characteristics of the individual layers.

The peer arrival process is a Poisson process with a mean inter arrival time of 5s. The peers are distributed among the classes according to a pre-defined share, by default half of the peers per class. Peers stay online until they have finished watching the video plus an exponentially distributed serving time with a default mean of 10min. Adding the buffering time of 60s before starting the playback and the video length of 22min, we thus get a mean number of concurrently online peers of ca. 400, neglecting stalling times.

TABLE I
Movie layer information

layer index	mean bit rate (kbps)	mean frame rate (fps)	cumulative mean frame rate (fps)	size (MB)
0	229	5.994	5.994	37.8
1	48	5.994	11.988	8.0
2	59	11.988	23.976	9.7

Additionally to the peers, we have a number of servers in the network that act like normal peers but have the complete video from the start and do not go offline during the simulation. A single server has an upload capacity of 512kbps, allowing us to scale the total server capacity by adjusting the number of servers. A high number of servers reflects the fact that a high capacity is provided by the content distributor to support the network. By default, we install a number of 40 of these servers.

Our main performance indicator for the evaluation is the end user Quality of Experience (QoE) when watching the video. As QoE indicators, we use two values we measure for each user. The first is the average number of layers played out over time. This allows us to see how many layers could be downloaded in time for playout. Since we use stalling for the base layer only, this value is always above 1 and below 3. The second performance indicator is the average stalling time of the peers. This value tells us how long blocks from the base layer were unavailable for playback, which we consider worse than playing back the video in a lower quality.

V. Numerical Results

In this section, we present the results from the simulation experiments we conducted. All values shown are mean values per run averaged again over several simulation runs. They are shown with their 95% confidence intervals. Each run simulated a swarm in the steady state for 5 hours.

We tested the proposed architecture for different network load conditions, with load being defined as the download demand in comparison to the total upload capacity of the swarm. Also, we varied the composition of the peer set and tested alternative download and chunk selection strategies.

A. Influence of Network Load

First, we take a look at the relation between network resources and the achieved QoE. Additionally, we want to find out how hybrid a P2P VoD system needs to be, i.e., how many servers are needed, to provide a good quality to the end user. To this end, we vary the number of servers from 1 to 80, which directly translates to the resources a content provider would offer to support the distribution effort. The resulting QoE indicators for the two peer classes are depicted in Fig. 4 and 5.

The first effect that can be observed is that a larger number of servers leads to a better quality played out at the peers, i.e., more peers can play out more layers on average. This is to be expected, since more servers simply mean a higher upload capacity in the network without adding download demand. Thus, the load in the system is reduced. However, this also shows that our adapted chunk selection can make use of the offered additional capacity by adapting the number of enhancement layers downloaded in time for playback. With 60 or more servers, i.e., 30Mbps base upload rate, the quality of the video is good or nearly perfect for all peers. In relation to the roughly 90Mbps total bandwidth demand by the clients, this shows that the overlay can handle a large part of the traffic load.

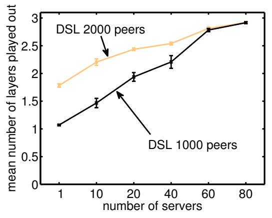

Fig. 4. Mean number of layers played out for different number of servers

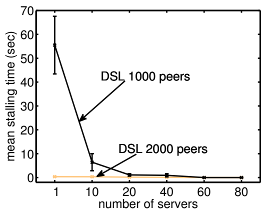

Fig. 5. Mean stalling times for different number of servers

Another interesting aspect of the architecture is that the average number of layers played out is higher for the set of DSL2000 peers. Also, these peers experience no stalling time, whereas the peers with less upload capacity show a high mean stalling time for a low number of servers. This can be explained with the G2G mechanism implemented in Tribler in the unchoking process. This mechanism prefers overlay neighbors with a good upload behavior in the unchoking, i.e., uploading, process of a local peer. Since peers with more capacity can upload more chunks, they get a better G2G rating and are therefore preferred in the unchoking process. For the P2P VoD application, this means that peers can actually influence the quality they get by adapting the upload capacity they allocate to the application, in case they do not utilize their full upload capacity. This is a good incentive for peers to contribute to the overlay.

B. Peer Heterogeneity and Server Breakdown

Next, we consider scenarios where we do not only vary the load by changing the share of the different peer classes, but we also let half of the default 40 servers fail simultaneously after half of the steady-state simulation time. Thus, we want to see how the chunk selection process reacts to the different load conditions, without any parameters being changed. Fig. 6 and 7 show the results. The x-axis denotes the share of DSL1000 peers to DSL2000 peers as, e.g., 10/90 if 10% of the peers are DSL1000 peers and the rest DSL2000 peers.

We see again that the peers with a higher upload capacity have a better QoE on average than their counterparts with less capacity. They can play out more layers more often, and experience a much shorter mean stalling time. We also can observe the effect of changing the peer set composition to include a higher share of peers with less upload capacity. Since this effectively means reducing the total upload capacity available to the system while keeping the download demand constant, a reduction in the QoE is to be expected and can be observed.

However, the DSL1000 peers suffer much more from this reduction of resources than the DSL2000 peers. While the DSL2000 peers experience a slight decrease in quality only, the average number of layers played out drops noticeably for the DSL1000 peers. Also, the mean stalling times increase drastically. This effect is exacerbated after half of the servers have failed, since then even less upload capacity is available in the network.

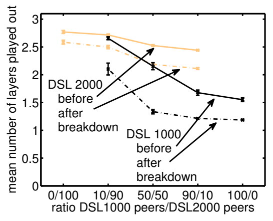

Fig. 6. Mean number of layers played out for different peer set compositions, before and after server breakdown

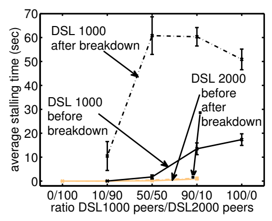

Fig. 7. Mean stalling times for different peer set compositions, before and after breakdown

In general, however, the chunk selection strategy adapted for SVC copes with the node failures and according load increase during the swarm lifetime without any parameters needed to be adapted to the new situation.

C. Influence of Download Strategy

The variants of the download strategy we want to compare in this experiment is the default strategy to stop downloading chunks when the client has finished playing out the movie, and a strategy where the peer continues to download the complete movie in any case. This means that a client consumes upload capacity of the swarm even after it does not profit from it, while other peers might put this capacity to better use since they are still watching the movie. The peer continuing to download may however be able to provide more content after it completes the movie file.

The results depicted in Fig. 8 and 9 show that this consideration does not pay off for the default setting of the serving time (10min). If the peers continue to download after they have finished watching the movie, all peers on average play out less layers, and the DSL1000 peers have a much higher stalling time. With this setting, the upload capacity consumption of the still downloading peers offsets the gain by having more sources for the complete video file. This should change with much larger serving times, however, which will be evaluated in future work.

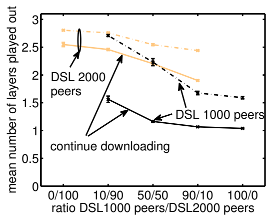

Fig. 8. Mean number of layers played out for different download strategies

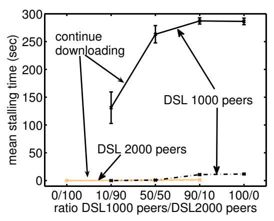

Fig. 9. Mean stalling times for different download strategies

D. Influence of Serving Times and Chunk Selection Strategy

In this experiment, we influence the serving time of the peers. Similar to the seeding time, a longer serving time leads to a higher available upload capacity and therefore less load. We vary the serving times from 5 to 30min. Additionally, we also want to compare variants of the chunk selection strategy with this experiment. For the chunk selection, we have the default implementation that prioritizes chunks from enhancement layers when being unchoked by a seeder. We compare this with a naive implementation that does not discern between seeders and leechers to see whether this mechanism has an effect.

The results (cf. Fig. 10 and 11) show that the naive chunk selection generally leads to a lower average number of layers played out for both classes of peers. However, this difference diminishes if the load in the network decreases with longer serving times.

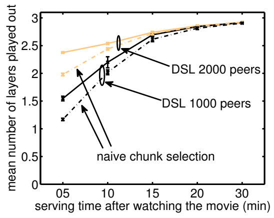

Fig. 10. Mean number of layers played out for different chunk selection strategies

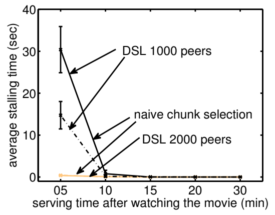

Fig. 11. Mean stalling times for different chunk selection strategies

On the other hand, the stalling times for the DSL1000 peers are higher with the more sophisticated chunk selection strategy in the scenarios with a short serving time. This can be attributed to the fact that base layer chunks are shared more with the naive chunk selection, and therefore are missing less often when they are required for playback. Thus, in a swarm with a high load, it does not necessarily pay off for all peers to prefer to download enhancement layers, even if the opportunity to do so appears seldom.

VI. Conclusion

In this work, we presented a P2P VoD architecture based on Tribler that supports streaming a SVC video. We extended the chunk selection strategy in a straightforward way to accommodate the layered structure of the video file. Since Tribler is a deployed and used application, this means our approach is realistic and can be implemented in a live P2P VoD system.

Our performance evaluation showed that the proposed strategy is able to adapt to the system load and peer access capabilities without having to measure network conditions or relying on feedback from the video player software. Additionally, it conserves and even reinforces incentives generated by the G2G peer selection to provide resources to the overlay. We compared different variants of the algorithms, providing an insight into the different situations where they are of advantage.

Interesting aspects that require future work are the addition of the two remaining scalability dimensions, creating a 'multi-dimensional' chunk selection. This more complex strategy then also needs to prioritize the type of quality that may be improved by downloading more layers from one of the scalability dimensions. Also, we want to take a closer look at the continuity of the quality that is received by the peers, since frequent changes in quality are not desirable for a good end user QoE.

Acknowledgments

This work has been conducted in the project G-Lab, funded by the German Ministry of Educations and Research (Förderkennzeichen 01 BK 0800, G-Lab). The authors alone are responsible for the content of the paper.

References

[1] Ipoque - Internet Study 2008/2009, "http://www.ipoque.com/," 2009.

[2] Garbacki P. Wang J. Bakker A. Yang J. Iosup A. Epema D. Reinders M. Van Steen M. Pouwelse, J. and H. Sips, "Tribler: A social-based peer-to-peer system," in *In The 5th International Workshop on Peer-to-Peer Systems (IPTPS'06)*, 2006, pp. 1–6.

[3] Engadget, "Nvidia unveils tegra devices," http://www.engadget.com/2009/06/02/nvidia-unveils-12-tegra-powered-devices-claims-the-mobile-comp/, 2009.

[4] ITU-T Rec. & ISO/IEC 14496-10 AVC, , "Advanced Video Coding for Generic Audiovisual Services," 2007.

[5] Marpe, D. and Wiegand, T. and Sullivan, GJ, "The H. 264/MPEG4 advanced video coding standard and its applications," *IEEE Communications Magazine*, vol. 44, no. 8, pp. 134–143, 2006.

[6] Marpe D. Wiegand T. Schwarz, H., "Overview of the scalable video coding extension of the h.264/avc standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1129, 2007.

[7] Dana, C. and Li, D. and Harrison, D. and Chuah, C.N., "BASS: BitTorrent assisted streaming system for video-on-demand," in *Proceedings of IEEE 7th Workshop on Multimedia Signal Processing*, 2005, pp. 1–4.

[8] A. Vlavianos, M. Iliofotou, and M. Faloutsos, "BiToS: Enhancing BitTorrent for supporting streaming applications," in *INFOCOM 2006. 25th IEEE International Conference on Computer Communications. Proceedings*, 2006, pp. 1–6.

[9] Mol, J., Pouwelse, J., Meulpolder, M., Epema, D., and Sips, H., "Give-to-Get: free-riding resilient video-on-demand in P2P systems [6818-03]," in *PROCEEDINGS-SPIE THE INTERNATIONAL SOCIETY FOR OPTICAL ENGINEERING*. International Society for Optical Engineering; 1999, 2008, vol. 6818, p. 6818.

[10] Baccichet, P., Schierl, T., Wiegand, T., and Girod, B., "Low-delay peer-to-peer streaming using scalable video coding," *Packet Video 2007*, pp. 173–181, November 2007.

[11] Zao J.K. Peng W. Hu C. Lin, C., H. Chen, and C. Yang, "Bandwidth Efficient Video Streaming Based Upon Multipath SVC Multicasting," in *Wireless Communications and Mobile Computing Conference*, August 2008, vol. 08. International Volume, pp. 406–412.

[12] Shen Y. Panwar S. Ross K. Liu, Z. and Y. Wang, "Efficient substream encoding and transmission for P2P video on demand," *Packet Video 2007*, pp. 143–152, 2007.

[13] Arnaud Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, New York, NY, USA, 2006, pp. 203–216, ACM.

[14] Joint Video Team (JVT) , "JMVM (Joint Multiview Video Model) software for the Multiview Video Coding (MVC) project of the Joint Video Team (JVT) of the ISO/IEC Moving Pictures Experts Group (MPEG) and the ITU-T Video Coding Experts Group (VCEG)," 2009.