Design and Evaluation of Tile Selection Algorithms for Tiled HTTP Adaptive Streaming

Jens Devloo[1], Nils Lamot[1], Jelle van Campen[1], Evi Weymaere[1], Steven Latré[1], Jeroen Famaey[1], Ray Van Brandenburg[2], and Filip De Turck[1]

[1] Ghent University – iMinds, Department of Information Technology, Gaston Crommenlaan 8/201, B-9050, Gent, Belgium
jens.devloo@ugent.be

[2] TNO, The Netherlands

Abstract. The future of digital video is envisioned to have an increase in both resolution and interactivity. New resolutions like 8k UHDTV are up to 16 times as big in number of pixels compared to current HD video. Interactivity includes the possibility to zoom and pan around in video. We examine Tiled HTTP Adaptive Streaming (TAS) as a technique for supporting these trends and allowing them to be implemented on conventional Internet infrastructure. In this article, we propose three tile selection algorithms, for different use cases (e.g., zooming, panning). A performance evaluation of these algorithms on a TAS testbed, shows that they lead to better bandwidth utilization, higher static Region of Interest (ROI) video quality and higher video quality while manipulating the ROI. We show that we can transmit video at resolutions up to four times larger than existing algorithms during bandwidth drops, which results in a higher quality viewing experience. We can also increase the video quality by up to 40 percent in interactive video, during panning or zooming.

Keywords: HTTP Adaptive Streaming, Tiled HTTP Adaptive Streaming, Client quality selection algorithms.

1 Introduction

Modern digital video is being created at resolutions much higher than several years ago. Standards like 4K UHDTV (Ultra High Definition Television) and 8K UHDTV are being more and more used and considered one of the important evolutions in future broadcasting. With the Internet being the predominant mean of delivering this content, transmitting video at high resolution in high quality demands for substantial amounts of bandwidth. Additionally, there is a demand for more interactivity in video. Concerts and sports games are filmed at big panoramic resolutions. The user is able to zoom in on a specific section. This means there is no need to transmit the entire video but users are only interested in a specific region of interest (ROI).

New techniques are being developed to counteract both the bandwidth problem, as well as to enable efficient delivery of interactive video. These techniques try to maintain the high quality and resolution of the source video, while allowing transmission across conventional Internet infrastructure. In our research, we evaluate the Tiled HTTP Adaptive Streaming (TAS) principle, a technique that attempts to minimize bandwidth costs by subdividing a video segment both spatially and temporally and encoding each segment in multiple quality levels. This is in contrast to HTTP Adaptive Streaming (HAS), where content is only segmented temporally. In TAS, we call these segments tiles and a tile selection algorithm needs to decide which tiles to download, i.e., which region of the video and in which quality. Given a fixed amount of available bandwidth, the clients' decisions can range from downloading a high-resolution region in lower quality video up to downloading a low-resolution region in the highest quality possible. Obviously, combinations of the above choices may exist and the client selection algorithm should take into account typical changes in the video's ROI by panning, zooming, etc. In this article, we propose three tile selection algorithms that correspond to different operations of the TAS technique. These include: scaled down video, cropped video, pannable video and pannable and zoomable video. An overview of the different supported operations can be found in Figure 1. We evaluate the performance of these situations using objective quality measures.

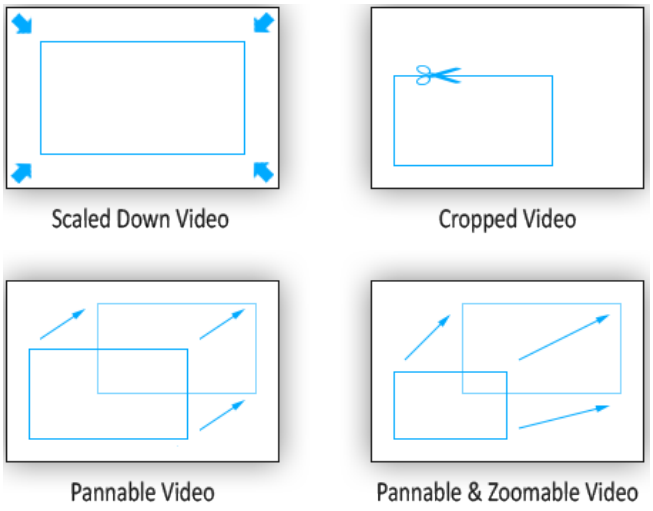

Fig. 1. Evaluated use cases for Tiled HTTP Adaptive Streaming (TAS)

The remainder of this paper is structured as follows. Section 2 describes related work in the field of HTTP Adaptive Streaming for high-resolution cinema. Section 3 explains the general principle of TAS. The three tile selection

algorithms are proposed and evaluated in Section 4 and 5, respectively. Finally, concluding remarks are given in Section 6.

2 Related Work

HAS is the third generation of HTTP-based streaming solutions. Several HAS techniques have been proposed by industrial players and standardization bodies, such as ISS Smooth Streaming (Microsoft) [1], HTTP Live Streaming (Apple) [2], HTTP Dynamic Streaming (Adobe) [3] and Dynamic Adaptive Streaming over HTTP (MPEG) [4]. Although differences exist in their details, all protocols exhibit the same set of architectural components. At the encoding side, the video content is first encoded in several different quality levels and resolutions. This is followed by a division of the content into temporal segments (typically several seconds worth of video) by a stream segmenter. These segments can be transported as a single file over HTTP. For each quality level, the most recently generated video segments are documented in a manifest file. This file also holds additional segment information such as segment location, size and quality. As such, the various segment files are linked into one video sequence through the meta-data contained in the manifest files. The segments and manifest files are hosted on one or more media distribution servers, typically HTTP web servers. Based on the information contained in the manifest files, the clients request the appropriate media segments through HTTP GET-methods. The client can then decide to download higher or lower quality segments to ensure a seamless rate adaptation.

In HAS, a video selection heuristic contained in the video client is responsible for deciding which quality levels are to be downloaded. Several modifications to these video selection heuristics have been proposed in the past. These new heuristics either modify the heuristic to improve its applicability to a particular domain or exploit the advantages of SVC-based HAS. For example, Liu *et al* discuss a specific video client heuristic for CDNs [5], while Adzic *et al* present a specific client heuristic for mobile environments [6]. Schierl *et al* propose a generic algorithm for selecting the next video quality to download [7]. Our solution also proposes a novel client heuristic, but is specifically oriented towards streaming ultra high resolution videos.

Van Brandenburg *et al.* extended the concept of HAS to also include spatially segmented video for ultra high resolution video, called Tiled HTTP Adaptive Streaming (TAS) [8]. They describe different approaches for using ultra-high definition video and how to use this video in a theatre, home viewing and mobile situation. It focusses on the mobile use case and describes how users are able to interactively navigate and look around in these ultra-high resolution videos. They present TAS and a first version of a tile selection algorithm. This algorithm is compared with our presented algorithms in Section 5. A more detailed overview of TAS is given in Section 3.

Similarly, Khiem *et al.* [9] compare two basic techniques (tile-based and monolithic streaming) for enabling zooming and panning in high definition videos

with constraints on the bandwidth and compression efficiency. Monolithic streaming is a mechanism where the pre-encoded video is first analysed to discover the dependencies between the macroblocks. When a certain area is requested, only those macroblocks that are needed for decoding this area in the video, are sent. The authors describe the differences between the two techniques and show the disadvantages of both mechanisms. They show that monolithic streaming is more bandwidth-efficient. It also has a better compression efficiency. This however is highly dependent on the chosen parameters for motion vector length, tile size, slice size (because of the variable length encoding) Their research allows us to discover the bottlenecks of the tiled-streaming techniques and consider them in our own research.

3 Overview of Tiled HTTP Adaptive Streaming

Before exploring our work, we first provide an overview of the basic functionality of TAS. Within TAS, an ultrahigh resolution video is split up both spatially and temporally. Spatially, the screen is subdivided horizontally and vertically in a number of tiles. TAS imposes no restriction on the amount of tiles. Typically the screen is subdivided a grid of either two by two or three by three tiles to be able to maintain the same aspect ratio for all spatial segments. Temporally, all of these tiles are split up in segments of a set duration. Segments can have any length, but its default is 10 seconds. A set of tiles at a resolution and quality is called a representation layer. The result of this process is a large collection of many possible tiles, ranging across various representation layers. The duration is typically fixed across all representation layers.

TAS' client-server architecture is shown in Figure 2. The client consists of a command parser, tile requester and a tile combiner/display. The command parsers receives commands to pan and zoom the video. The tile requester requests individual tiles from the video. When determining what tiles to send, it takes in account the ROI and screen resolution of the playback device. The combiner then combines all received tiles into a single stream of video. After doing so it will display the resulting footage on screen.

The starting point of our research is a prototype TAS client-server infrastructure, developed in the context of the European FP7 FascinatE project. The prototype consists of a client, which contains a basic tile selection algorithm, which will be used as a reference to compare our algorithms. This tile section algorithm works as follows: the client requests the tiles that fall within the ROI at high quality, while also requesting the whole video at a minimal quality. These minimum quality tiles are called the fallback layer. This layer is required to allow for latency-free panning and zooming to areas outside of the ROI.

4 Tile Selection Algorithms

In this section, we give an overview of the algorithms developed for the four use cases considered in our research. We consider four different use cases: scaled

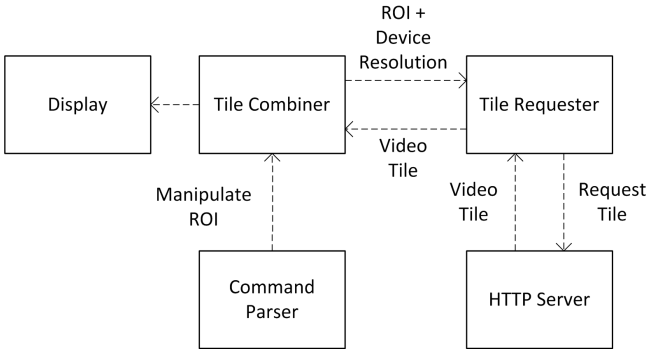

Fig. 2. Overview of the TAS' architecture

down video, cropped video, pannable video and pannable & zoomable video. We developed three tile requesting algorithms that cover these use cases. By scaled down video, we mean that all segments of the video are completely shown on the screen. No pixels fall outside of the view. We also use the entire resolution of the playback device to show the video. In this situation neither panning nor zooming is possible. Cropped video only shows a part of the video (spatially). This part will be displayed at the highest available resolution. We consider the size of this spatial part of the video equals the screen resolution of the display device. This situation allows the user to zoom in on a specific ROI, while panning is not possible. With the third use case, pannable video, only a certain part of the video (spatially) is shown. This part will also be displayed at the highest available resolution and the user will be able to pan around, thus changing what part of the video is displayed. In this situation no zooming is possible. At last, we combine the two previous use cases in a pannable and zoomable use case. The user will be able to perform both operations at the same time.

4.1 Scaled Down Video

As the entire video is visible at all times, all tiles will be shown at the same quality at all times. As the video is scaled down from the ultra high definition to the screen resolution, high quality details might not be visible, so we do not need to send the tiles at highest quality. This is shown in Algorithm 1: the algorithm selects the highest quality that is (i) required by the device's screen resolution and (ii) can be streamed through the network.

4.2 Cropped Video

This algorithm is shown in Algorithm 2 and extends the algorithm described above. It still selects the highest possible tiles in terms of quality to send, but also takes into account what users are actually viewing in their ROI. In this situation, panning is impossible. This implies that we do not need to send surrounding tiles and thus only send segments within the ROI.

```
determine quality based on screen resolution;
determine bandwidth needed for this quality;
while not enough bandwidth available do
    lower quality of all tiles by one;
end
send all tiles at determined quality;
```

Algorithm 1: Scaled Down Video Algorithm

```
determine quality based on screen resolution;
determine bandwidth needed for this quality;
if not enough bandwidth available then
    sort visible tiles on ascending visibility (visible area/total area);
    while not enough bandwidth available do
        remove least visible tile;
        if equivalent at lower layer not already in list then
            replace tile by equivalent at lower layer;
        end
    end
end
send all visible tiles at their determined quality;
```

Algorithm 2: Cropped Video Algorithm

4.3 Pannable & Zoomable Video

As shown in Algorithm 3 and similar to the previous case, we opt to send all tiles inside the ROI at maximum possible quality. As panning operations need to be supported, all non-visible tiles are send at a lower quality. The algorithm assigns the highest quality to the ROI tiles and then assigns lower qualities to neighbouring tiles: the further the tile is away from the ROI, the less priority it has to receive a high quality. Additionally, as zooming is possible as well, we use the remaining bandwidth to increase the quality of the visible segments.

5 Evaluation

5.1 Experimental setup

The above described algorithms were evaluated on the iMinds iLab.t Virtual Wall testbed facility, which is a large scale testbed facility for setting up network experiments. We modelled a client server topology, with a network link. By default, the capacity of this link is 50 Mbps, but in some tests this was varied to introduce additional bottleneck scenarios. To evaluate the several algorithms, we focus on two different metrics:

– **Bandwidth used for visible video**: the amount of bandwidth dedicated to video that is actually being shown on the client device.

```
determine quality based on screen resolution;
determine minimum quality for non-visible tiles;
determine bandwidth needed for these qualities;
create a list with the needed non-visible tiles surrounding the ROI at minimum
  quality;
if not enough bandwidth available then
    sort visible tiles on ascending visibility (visible area/total area);
    while not enough bandwidth available do
        remove least visible tile;
        if equivalent at lower layer not already in visible or non-visible tiles
        then
            replace tile by equivalent at lower layer;
        end
    end
else
    sort non-visible tiles on ascending ROI-distance;
    while enough bandwidth available do
        remove closest non-visible tile;
        if equivalent at higher layer not already in visible or non-visible tiles
        then
            replace tile by equivalent at higher layer;
        end
    end
end
send all tiles at their respective quality;
```

Algorithm 3: Pannable & Zoomable Video Algorithm

– **Overall visible quality**: a measure of the visual quality that is shown to the user. This is calculated as the sum of the separate quality values (q_i) of the tiles multiplied by the percentage of the visible area this tiles fills (p_i). We make use of the number of the presentation layer (corresponding with a Mean Opinion Score of the video) as the quality value for a certain layer.

$$Overall\ video\ quality = \sum_{i=1}^{N} q_i * p_i \tag{1}$$

Here, N corresponds to the number of segments of a video. The test video footage we use in our experiments consists of a four minutes and 40 seconds long video, split across 10780 files. These files encompass 28 time intervals (temporal division) and 10 representation layers. The number of the representation layer corresponds with the amount of horizontal and vertical tiles (e.g. quality layer 4 contains tiles for a 5 by 5 divisions, quality layer 5 contains tiles for a 6 by 6 divisions). All of these videos have the same resolution. This means, the smaller an area the tile describes, the more detailed the total video is stored. Representation layer 0 is made up of a single tile, stored at 408 by 174 pixels. Representation layer 5 consists of 36 tiles (six horizontally by six vertically) all

stored at 408 by 174 pixels. Thus the video quality and overall spatial resolution will be higher for higher representation layers.

5.2 Results description

Standard TAS algorithm The standard TAS algorithm requests two streams for every tile. A fallback segment with the lowest quality at a highest priority and a viewable segment of a chosen quality at a normal priority. The quality of the viewable segment depends on the zooming factor and on the amount of tiles that were specified in the configuration file. The fallback segment is for backup purposes, it will be used when a necessary standard segment could not be downloaded in time or when the user zooms or pans so the necessary standard segment is not available yet. This solution is very bandwidth friendly, because the fallback layer has a very low quality, and thus requires a low bitrate.

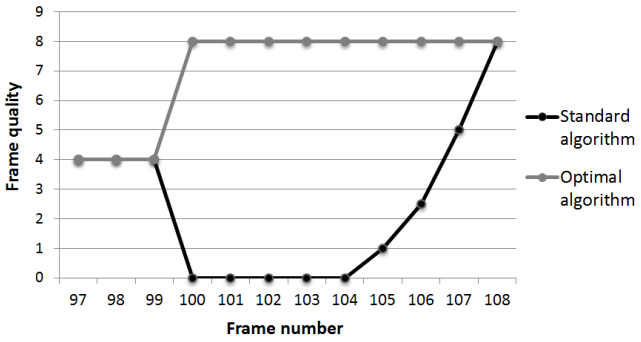

Fig. 3. Segment qualities after a zoom operation

Figure 3 shows the result of a zoom operation just before video segment 100 will be shown on the screen. In this experiment, we zoomed from a 4x4 tiles perspective with quality 4 to a specific region that exactly covers 16 segments (4 by 4) of quality 8. The algorithm does not have the necessary tiles available and falls back to the lowest possible quality and playbacks the lowest quality. This lasts a couples of segments until the correct segments are requested and have arrived. We see the segment quality rise to 1 in segment 105. This means the algorithm now uses two layer 8 segments and 14 segments from the fallback layer. An optimal algorithm, in terms of quality, should immediately start using 16 segments from segment layer 8 and achieve an optimal video quality of 8 as shown in Figure 3. This is impossible to do in practice without having to download all segments in all possible qualities. The algorithm proposed in this paper, have a performance which is located between those two extremes.

Scaled Down Video Algorithm To evaluate the scaled down video algorithm, the bandwidth was gradually lowered from 50 Mbps (enough for downloading

all tiles at the highest possible quality) to 1 Mbps (only enough to download the fallback layer).The results are shown in Figure 4: when the bandwidth is not sufficient to download the tiles at the defined quality, the standard algorithm uses the fallback layer. Our algorithm calculates which quality can be achieved with this bandwidth, which results in a more efficient use of the available bandwidth. The proposed algorithm thus delivers segments at a higher quality.

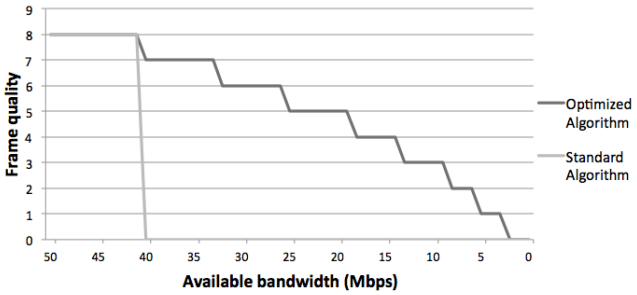

Fig. 4. Segment quality versus decreasing bandwidth

Cropped Video Algorithm To evaluate the cropped video algorithm, we simulated a bandwidth drop in segment 103 and segment 108. Figure 5 shows that the standard algorithm is not able to download all necessary tiles of quality 5 after the bandwidth drop and uses a scaled version of the fallback layer instead. Our proposed algorithm reacts on the bandwidth drop, calculates the maximum possible quality with this new bandwidth and requests the needed segments. So when a bandwidth drop occurs, the quality lowers to the highest possible quality that is still retrievable in time. When simulating another bandwidth drop at segment 108, we observe no difference for the standard algorithm, as it is already downloading and showing tiles at the lowest quality. Our algorithm will again lower its quality to the highest possible quality at the new bandwidth.

As discussed, our algorithm reacts much better on bandwidth variation than the standard algorithm. In Figure 6 we can observe the used bandwidth for both algorithms. Both algorithms use around 18 Mbps when showing all 25 tiles and when no bandwidth has been dropped. The fallback layer only needs 0.5 Mbps, as it consists of only one tile. As expected, the algorithm uses the available bandwidth much better, but at a significantly lower quality. Also, if we lower the bandwidth to a level where only the fallback layer can be downloaded in time, the standard algorithm and our proposed algorithm yield the same results.

Pannable & Zoomable Video Algorithm For the evaluation of the pannable & zoomable algorithm, we start with an 8x8 tiled video and zoom in to a ROI that is constructed by exactly 25 (5x5) of the 64 tiles. We evaluate our algorithm

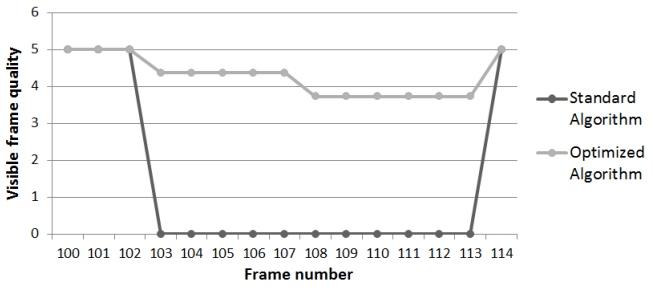

Fig. 5. Segment qualities after bandwidth drops

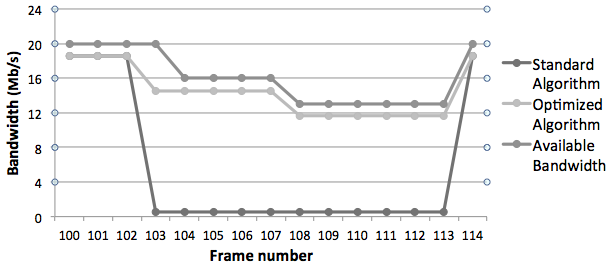

Fig. 6. Necessary bandwidth for the standard and our algorithm

by panning a tile to the right at segment 103 and again at segment 106. Figure 7 shows the resulting visible quality during this operation. The standard algorithm will fill in the five tiles on the right border with a scaled up version of the fallback layer. As such, the overal quality drops to 4 after the initial panning step. In contrast, our algorithm has already downloaded the surrounding blocks at the highest possible quality with the available bandwidth. In this scenario, our algorithm is able to fill in the five tiles at the right with quality 3 segments instead of the fallback layer. This results in the following quality:

$$\sum_{i=1}^{N} q_i * p_i = 20 * 5 * \frac{1}{25} + 5 * 3 * \frac{1}{25} = 4.6 \tag{2}$$

A second panning command (at segment 106) again results in a lower quality. The quality of a segment with our algorithm decreases to 4.2, while the quality of a segment with the standard algorithm decreases to 3. Assuming the requesting and downloading of the correct tiles takes 5 segments, it lasts until segment 108 for the quality to increase again. As also shown in Figure 7 this algorithm is already close to the optimal version, which assumes infinite bandwidth.

A comparison of the used bandwidth is given in Table 1. As shown, our algorithm uses more bandwidth than the standard algorithm. This is the result of continuously downloading surrounding blocks on top of the tiles within the

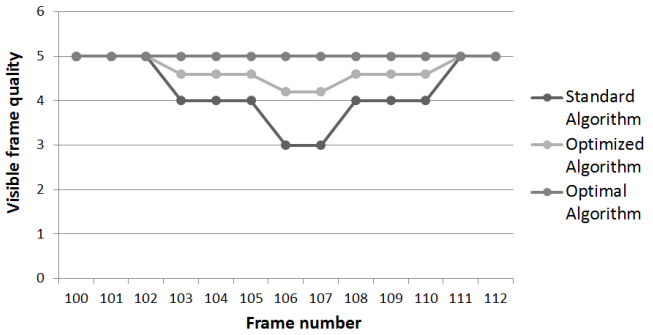

Fig. 7. Segment qualities after panning

Bandwidth Used with	Mbps
Standard Algorithm	5.01
Our Algorithm	9.66
Optimal algorithm	12.55

Table 1. Bandwidth used

ROI. The standard algorithm only downloads the tiles within the ROI and the fallback layer as backup. However, as such, it is better in utilizing the available network capacity to maximize the video quality.

6 Conclusion

In this article, we presented three different tile requesting algorithms, corresponding with different TAS use cases. Compared to a standard TAS algorithm, each of the algorithms takes a more intelligent decision on user operations such as zooming, panning or a drop in bandwidth, to maximize the video quality at a higher bandwidth. These algorithms were evaluated and compared to the standard TAS algorithm. The results show that the proposed tile requesting algorithms can obtain much higher quality than the standard algorithm in their respective use-cases. By utilizing the available bandwidth in an efficient way, we deliver video at a much better quality. We also optimized the tile requesting algorithms for different use-cases, so the best user experience can be achieved for each use-case. In case of pannable video, there is a trade-off between the quality of visible tiles and the quality of non-visible tiles when the bandwidth is not sufficient. By making it possible to change the lowest allowed quality of non-visible tiles, we gave the opportunity to work with profiles for different forms of user interaction. This way, if the user pans a lot, the profile can require a higher minimum quality of non-visible tiles than a profile where the user only pans occasionally. In further research, we plan to search ways to monitor the behaviour of the user and change the profile dynamically.

References

1. Microsoft: Microsoft smooth streaming http://www.iis.net/downloads/microsoft/smooth-streaming - Last accessed on February 14, 2013.
2. Pantos, R., May, W.: Http live streaming. Internet-Draft draft-pantos-http-live-streaming-10, IETF Secretariat (2012)
3. Adobe: HTTP dynamic streaming - high-quality, network-efficient http streaming http://www.adobe.com/be_nl/products/hds-dynamic-streaming.html - Last accessed on February 14, 2013.
4. Stockhammer, T.: Dynamic adaptive streaming over HTTP – standards and design principles. In: Second annual ACM conference on Multimedia systems. (2011) 133–144
5. Liu, C., Bouazizi, I., Hannuksela, M.M., Gabbouj, M.: Rate adaptation for dynamic adaptive streaming over http in content distribution network. Signal Processing: Image Communication **27**(4) (2012) 288 – 311 ¡ce:title¿Modern Media Transport – Dynamic Adaptive Streaming over HTTP (DASH)¡/ce:title¿.
6. Adzic, V., Kalva, H., Furht, B.: Optimized adaptive http streaming for mobile devices. In Tescher, A.G., ed.: Applications of Digital Image Processing XXXIV. Volume 8135., SPIE, SPIE (2011) 81350T
7. Schierl, T., Sanchez de la Fuente, Y., Globisch, R., Hellge, C., Wiegand, T.: Priority-based media delivery using svc with rtp and http streaming. Multimedia Tools and Applications **55** (2011) 227–246 10.1007/s11042-010-0572-5.
8. van Brandenburg, R., Niamut, O., Prins, M., Stokking, H.: Spatial segmentation for immersive media delivery. Intelligence in Next Generation Networks (ICIN) (2011)
9. Khiem, N.Q.M., Ravindra, G., Carlier, A., Ooi, W.T.: Supporting zoomable video streams with dynamic region-of-interest cropping. Proceedings of the first annual ACM SIGMM conference on Multimedia systems (2010)