BITRATE AND VIDEO QUALITY PLANNING FOR MOBILE STREAMING SCENARIOS USING A GPS-BASED BANDWIDTH LOOKUP SERVICE

Haakon Riiser[1]*, Paul Vigmostad*[1]*, Carsten Griwodz*[2,3]*, Pål Halvorsen*[2,3]

[1]Netview Technology AS, Norway [2]Simula Research Laboratory, Norway [3]Department of Informatics, University of Oslo, Norway

ABSTRACT

One of the main challenges when streaming video to mobile devices is to handle fluctuating bandwidth and frequent network outages as the device is brought in and out of areas with network coverage. In this paper, we propose bitrate and video quality planning algorithms for mobile streaming scenarios using a *GPS-based bandwidth-lookup service* in order to reduce frequently changing video quality and the number of playout interruptions. Our real world experiments using an adaptive segmented HTTP video streaming system while travelling on popular commute routes indicate that the users' quality of experience is greatly increased.

Index Terms— adaptive streaming, GPS-based bandwidth prediction, bitrate and quality planning

1. INTRODUCTION

A large part of the Internet traffic today originates from video services, and the number of streaming services to mobile wireless devices is increasing. For example, commuters on buses, trains, trams, boats, etc. are streaming videos to devices such as smart phones, tablet PCs and laptops to watch the latest news or sports updates. Unfortunately, it is a well known fact [11] that wireless networks suffer from fluctuating bandwidth and frequent network outages as the device is brought in and out of areas with network coverage.

Current developments in adaptive streaming have made it possible to change bitrates as the network bandwidth changes, e.g., using a scalable codec [3, 13, 4] or an adaptive HTTP streaming approach [7, 15, 10]. Using these technologies, the bitrate (and thus video quality) can be changed dynamically to match the oscillating bandwidth, giving a large advantage over non-adaptive streams that are frequently interrupted due to buffer underruns or data loss.

Still, adaptive video formats and streaming systems only offer the means to change the quality. The applications must determine how to use these quality adaptation mechanisms. Choosing the right video bitrate at the right time, and knowing how much to buffer, are non-trivial tasks. For example, the quality should preferably not change too often, nor should it change by too many levels in a single jump as it negatively affects the user's perceived quality [9, 16]. Ideally, one should trade off quality for more buffering to hide connectivity gaps. Current quality adaption methods do not plan ahead, causing frequently changing quality and buffer underruns. Consequently, the users' quality of experience (QoE) is degraded.

The easiest option is to use a low video bitrate and download as fast as possible until the buffer is large enough to last through most normal outages. This conservative approach means that the quality is often much lower than what would be possible if all fluctuations in bandwidth could be foreseen, and ideally, the receiver would know in advance precisely how the network behaves over the course the streaming session. To enable smart pre-buffering and smooth video quality over long periods of time, bandwidth prediction is essential.

To support bitrate and video quality planning in a video streaming to mobile devices scenario, we propose a *GPS-based bandwidth-lookup service*. This service allows video receivers equipped with a GPS to help build a database that enables bandwidth prediction using geographical location and past network conditions. Furthermore, we present bitrate and video quality planning algorithms that avoid buffer underruns during outages and reduce the number of quality changes. We experimentally evaluate our system through both simulations and real-world tests using an adaptive media player [8] along different commute routes. Our results indicate that the video quality is much more stable and the number of playout interruptions is drastically reduced.

2. RELATED WORK

While video streaming to mobile devices has been a hot research topic for several years, a remaining challenge is to adapt video streaming to the unpredictable behavior of wireless networks such as General Packet Radio Service (GPRS) and High-Speed Packet Access (HSPA). In these networks, fluctuating network bandwidths strongly influences the video streaming performance [2].

An important strategy for coping with the effects of bandwidth variations, is video bitrate adaption. Currently, several commercial systems like Apple's HTTP Live Streaming [10] and Microsoft's Smooth Streaming [15] monitor the download speed of video segments and adapt dynamically to resource availability changes by switching between video segments coded in different qualities and bitrates. Several similar

This work is funded by the Norwegian Research Council: the *HyStream* project (176847) and the *iAD* centre for Research-based Innovation (174867).

preload approaches use (hierarchical, layered) scalable video codecs like scalable MPEG (SPEG) [4], Multiple Description Coding (MDC) [3] and the Scalable Video Coding (SVC) extension to H.264 [13] to adapt the quality by filling a prefetch window bottom-up, i.e., starting with the lowest quality and building up the layers until the window is played out [4, 12]. These are popular approaches, but potential challenges are too frequent quality switches which may annoy the users [16, 9], and buffer underruns due to connection outages.

Thus, there is a need for orthogonal mechanisms that predict connection losses and bandwidth fluctuations to adapt better. A general QoS-prediction service has been suggested by Wac et al. [14], and like our bandwidth prediction service, they suggest that the users of the database could be responsible for filling it. How a specific application should use this QoS-prediction service is outside the paper's scope. Furthermore, a "gap filler" is proposed in [6] which predicts gaps by observing the signal behavior and parameters like propagating mode, the form of the tunnel, attenuation due to roughness of the walls and antenna insertion loss. A related idea of a geo-predictive media delivery system has recently been developed in parallel to our work by Curcio et al. [1]. Their system predicts network outages and prebuffers video. However, these works provide only simulation results, do not use adaptation, and do not consider mechanisms for route planning. In contrast, our results are based on a running prototype with real-life experiments (complemented by simulations) that stream video to a mobile device whose user travels along popular commute routes in Oslo.

3. VIDEO STREAMING WITH A GPS-BASED BANDWIDTH-LOOKUP SERVICE

The number of video streaming services to mobile devices is increasing, but one of the challenges that must be addressed is the variable network connectivity with fluctuating bandwidth availability and the many network outages. In this respect, our experiments have shown that the network variability often is related to the geographical position. We have implemented a GPS-based bandwidth-lookup service to stream video with a more stable quality and to enable buffering for periods of connection loss.

The basic idea is to monitor a video streaming session's data download rate and geographical location. By uploading this data to a central service, the users themselves help building a database that can provide an invaluable service to future users. Using such a lookup service, users can query for predicted available bandwidth for given locations based on the reported previous observations. For users commuting using public transportation[1], both the commute time and route are approximately known. This means that our proposed lookup service can be used to plan the bandwidth availability

[1]We are currently working on scenarios where the movement is less predictable and shorter prediction windows are used, but this is ongoing work and outside the scope of this paper.

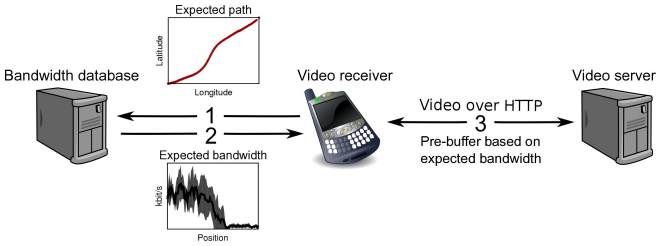


Fig. 1. System architecture.

throughout the whole trip filling potential gaps and maintaining a stable video quality during bandwidth oscillations.

3.1. Architecture and streaming operations

The architecture of our GPS-based bandwidth-lookup service is illustrated in figure 1, and during a streaming session, the client first looks up the predicted bandwidth along the route before starting the streaming session at a calculated quality. The following three steps are performed in our prototype:

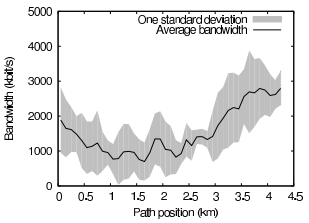
1. The receiver sends the commute route to the GPS-based bandwidth-lookup service, providing the latitude and longitude for every 100 meters along the path[2].
2. Once the lookup server receives the path description, it returns data for every point listed in the path description. The data for each point includes the average bandwidth, variation in bandwidth and average speed.
3. The receiver now has predictions for path, bandwidth and travel time. Using this plus the video quality levels' average bitrates, we calculate the estimated number of bytes that can be downloaded during the rest of the trip. This information is used to plan which quality levels to use; if the prediction is good, the effect of connectivity gaps is "smoothed" out over time.

3.2. Data sets

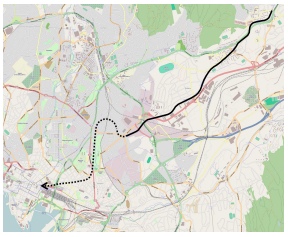
To build a data set based on real world measurements for our prototype, we have performed field trials along several popular commute routes to downtown Oslo, Norway. The general idea with the lookup service can be applied to any kind of network, but in our initial proof-of-concept prototype, we have used the available 3G mobile network in Oslo. The network is based on the Universal Mobile Telecommunications System (UMTS) architecture. The field experiments are performed using laptops with 3G mobile Internet devices (Huawei Model E1752 HSPA USB stick) and GPS (Haicom HI-204III USB GPS). The bandwidth measurements were conducted using an HTTP adaptive streaming system [5] that downloads video segments. Due to space restrictions, we present two route examples where each route has multiple

[2]Non-GPS clients can also use this information for planning trips using known routes like in a public transport scenario, although they cannot adapt to unexpected situations. This kind of use is beyond the scope of this paper.

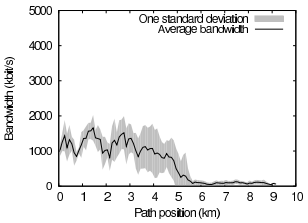
(a) Ferry commute path.



(b) Measured bandwidth along the ferry commute path.



(c) Metro commute path (the dotted parts are underground).



(d) Measured bandwidth along the metro commute path.

Fig. 2. Two example commute routes into downtown Oslo (Norway): *metro* between Kalbakken and Jernbanetorget, and and *ferry* between Nesodden and Aker Brygge.

measurements used to give the users of the service both average values and variance.

A popular way of commuting from the Nesodden peninsula to downtown Oslo is by boat. Because of poor signal conditions far from land, streaming is problematic. Figure 2(a) shows the route, and figure 2(b) presents the measured values with standard deviations. The available bandwidth varies quite a lot – both in terms of the average and the variance. We measured rates up to several Mbps and down to a few kbps. However, the signal is (usually) never completely gone while crossing the Oslofjord. As expected, the signal is strongest when the ferry is close to land (Nesodden at the start of the path, and Aker Brygge in downtown Oslo).

The other scenario described in this paper is commuting by metro in Oslo. The path is depicted in the map in figure 2(c), and the corresponding bandwidth measurements are presented in figure 2(d). From these experiments, we observe that the bandwidth is relatively good before entering the tunnel, but *very* poor inside the tunnel.

These observations mean that, with sufficient information about bandwidth availability before reaching mid-sea or entering the tunnels, a video streaming system with a smart bitrate planning algorithm can stream at a slightly lower rate when passing through high bandwidth areas, and use the remaining bandwidth to prepare for the low bandwidth areas further ahead on the commute path.

4. BITRATE AND VIDEO QUALITY PLANNING

When predicting the available bandwidth and planning how to best consume the available bandwidth, our goal is to make a bandwidth prediction as close as possible to the actual observed bandwidth. Our goal is to perform quality switching so that we consume the buffer completely while at the same time avoid buffer underruns during outages and reduce the number of quality changes. We have tested four different algorithms:

REACTIVE A *buffer-based reactive* algorithm that selects video bitrate based on the number of seconds of video that are preloaded in the input buffer, i.e., this is an algorithm similar to what HTTP streaming systems use without prediction. When the buffer is empty, the lowest quality level is selected. For every ten seconds of video in the buffer, the quality level increases by one level, until the maximum level is reached (the quality level decreases by the same rules when the buffer drains). This is simple to implement, but with this method, bandwidth peaks and dips have matching video quality peaks and dips. If the network bandwidth drops below the bitrate of the lowest quality level for a significant period of time, buffer underruns and playout interruptions are bound to happen.

MONTE-CARLO A history-based prediction algorithm, which we call *Monte-Carlo prediction*, predicts bandwidths using a Monte-Carlo method based on recorded values. This method generates N different hypothetical runs ($N = 2000$ in our simulations), where each run constructs a path where a random historic bandwidth measurement is picked for a 100 meter radius around the predicted location for every second of the predicted trip. These N runs are then sorted by the bitrate they could support without buffer underrun, and one is picked as our prediction. The one to pick depends on an "optimism" parameter (e.g., pick a poorly performing run for a pessimistic estimate). This parameter is necessary to avoid buffer underruns in streaming sessions that are steadily getting lower than average bandwidth. Real-world observations prove that this happens.

STDDEV A history-based prediction algorithm that predicts bandwidths using statistical distribution of stored bandwidth numbers in the database. This method creates a path where, for every 100 meters along the path, we pick a bandwidth that is p times the standard deviation (σ) higher than average at that position:

$$\text{predicted\_bw}(pos) = \text{average\_bw}(pos) + p \cdot \sigma(pos)$$

For example, picking $p = -0.5$, our predicted bandwidth numbers are half the standard deviation below than the average.

OMNISCIENT To compare our results to an optimal scenario, we also used an *omniscient prediction* algorithm. Unlike the above methods, this a posteriori algorithm

has all measurements for a particular test run available. This algorithm chooses bandwidths in such a way that it never needs to reduce quality, increases quality as few times as possible, and consumes its buffers completely.

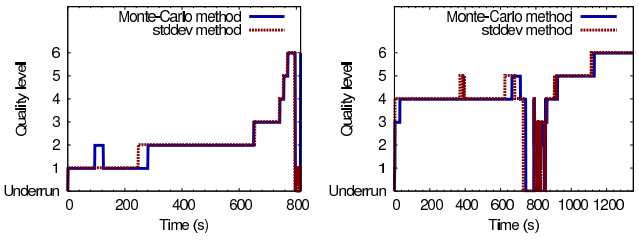
When estimating the bandwidth using the latter three prediction methods, we calculate the bitrates that we can support without getting a buffer underrun. In case of failure, our prediction system tries to adapt. The Monte-Carlo method then chooses a more pessimistic run, and the stddev method changes the p value. Our goal is to only go *up* in quality over time[3], and to have an empty buffer when we arrive at the end of the path. Finally, we want to achieve this with as few quality changes as possible in order to provide an optimal QoE.

5. EXPERIMENTS

We have tested our proposed solution in both a simulation environment based on real-world bandwidth traces and a real-world environment using an adaptive segmented HTTP streaming system [8] similar to systems like Apple's HTTP Live Streaming [10] and Microsoft's Smooth Streaming [15]. Here, the video player downloads video segments, and each segment is coded in multiple rates (and thus qualities) to enable quality adaption according to available bandwidth. For content, we used European football (soccer) matches encoded using two-second segments in six quality levels with bitrates of 250, 500, 750, 1000, 1500 and 3000 kbps.

5.1. Simulation

Performing real streaming experiments in a live, mobile environment for all configurations of the system is very time consuming. To find the best algorithms to use and a suitable configuration, we first performed a set of simulations. For a chosen bandwidth log (from our initial real measurements of network bandwidth), we simulated a media player which picks quality levels based on one of the algorithms described above. In this way, we can compare different quality selection algorithms on the same bandwidth trace and compare the results directly.



(a) Metro (b) Ferry

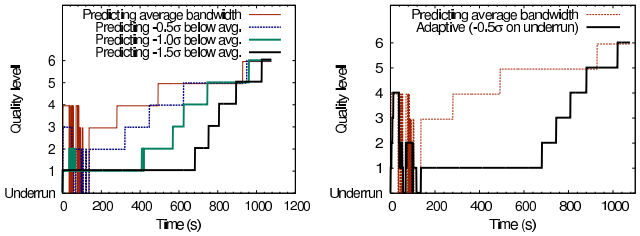
Fig. 3. Comparing the Monte-Carlo and stddev methods.

[3]There are several reasons, but our experience indicate that it is better to be a bit careful in the beginning and increase as you have built up a buffer. Nevertheless, this can easily be changed.

First, to evaluate and compare the two prediction algorithms, and choose one, we simulated their behavior when executed on one trace from each of the three commute routes. The results are shown in figure 3. We can observe that the Monte-Carlo method and the stddev method produce very similar results. In figure 3, both algorithms were configured to make a "neutral" prediction, but similar results can be shown for both pessimistic and optimistic aggressiveness. However, the Monte-Carlo prediction algorithm has the disadvantage of being very expensive to compute compared to the stddev method, so the rest of this study does not consider the Monte-Carlo method and only uses the computationally inexpensive stddev method.

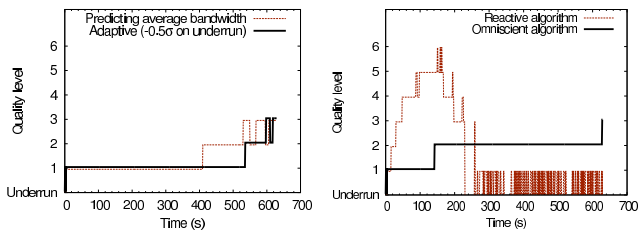
Our next experiments target the prediction aggressiveness. Our initial approach was to use a fixed aggressiveness setting, but the results from our simulations using a number of bandwidth logs showed that a fixed setting made the algorithms prone to buffer underrun unless a very conservative estimate was chosen, severely under-utilizing the bandwidth and resulting in a low overall video quality. Figure 4(a) shows the behavior of the stddev algorithm on a challenging bandwidth log where a buffer underrun (due to a large network outage) occurs at the beginning of the run. It shows that only the very pessimistic prediction of 1.5σ (standard deviation) below average manages to avoid buffer underruns. However, while such a pessimistic setting avoids buffer underruns, the video quality utilizes the available bandwidth only very rarely.

As a compromise, we designed the *adaptive stddev algorithm*; it adapts by lowering the p multiplier of σ on each buffer underrun. In addition, we also included parts of the reactive buffer algorithm; when choosing a quality level, we calculate quality levels from both the predictive model and the reactive algorithm. We then choose the lowest of these levels. This avoids trusting the prediction too much when the margins are small due to little data in the buffer. The algorithm starts with neutral assumptions, allowing for high quality when the network conditions are good, and gradually lowers expectations when the conditions are bad. Compared to the non-adaptive approach, it reduces the number of buffer underruns considerably.



(a) Four levels of aggressiveness. (b) The effect of adaptive.

Fig. 4. The effect of using adaptive aggressiveness (both plots use the same trace, which is a worst case scenario where the buffer underruns happen in the beginning).

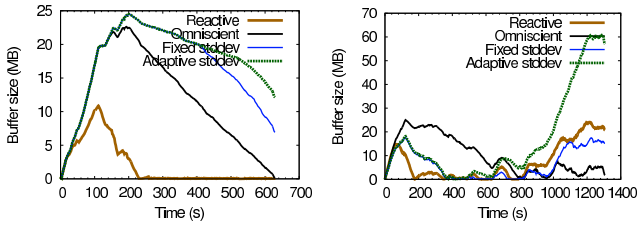


(a) Fixed stddev vs adaptive stddev (b) Reactive vs omniscient

Fig. 5. Comparing fixed stddev (neutral aggressivity), adaptive stddev, reactive and omniscient algorithms on metro.

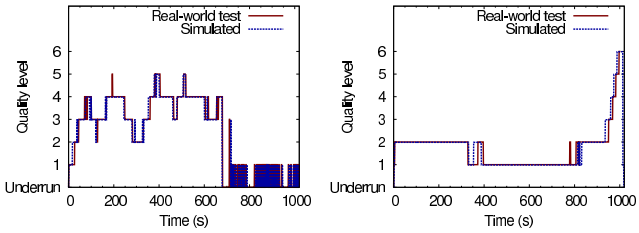
Figure 4(b) compares the stddev algorithm (neutral settings) with the adaptive stddev algorithm (set to lower prediction by 0.5σ on each buffer underrun). We found that this method serves as a good compromise between stability and average quality. The results in figure 4(b) show that three buffer underruns at the start reduce the prediction to 1.5σ below average, resulting in the pessimistic (but safe) prediction shown in figure 4(a). We also experimented with restoring the optimism over time, but this resulted in more buffer underruns, so for now, we chose the more conservative algorithm and reset the optimism only between streaming sessions (even though the scenario in figure 4(b) shows a potential for improvement).

Having decided on an algorithm, we present in figure 5 comparisons of the fixed stddev algorithm, the adaptive stddev algorithm, the reactive buffer algorithm and the omniscient algorithm using one of the metro-route logs (the other routes show the same trend). The first observation is that the traditional reactive algorithm has severe buffer underruns. For some experiments (not shown), the fixed prediction algorithm failed to provide a smooth video playback, but the adaptive prediction algorithm is able to adapt after the first buffer underrun and shows a similar behavior as the omniscient algorithm. Furthermore, for these three tests, we also present the buffer fullness in figure 6. From the plots, we can observe that the omniscient algorithm always ends with an empty buffer due to the perfect prediction, while the more conservative algorithms have more in the buffer at the trip's end. The reason for this is that they do not stop downloading video until the player is shut down, and the playout position is far behind the download position when the bandwidth prediction is below the actual bandwidth, i.e., the under-prediction is used to download video data beyond the estimated arrival time.



(a) Metro (b) Ferry

Fig. 6. Comparing buffer-fullness for the different algorithms.



(a) Reactive buffer algorithm (b) Neutral stddev algorithm

Fig. 7. Testing the accuracy of the simulations by comparing real-world tests of algorithms with the simulation (using the bandwidth measured in the real test to make them comparable)

Finally, to evaluate the accuracy of our simulations, we first ran a real streaming session using the Netview player and then ran a simulation over the *same* monitored bandwidth log. The results with respect to the quality level as a function of time are shown in figure 7. We see that the real and the simulated systems make almost exactly the same decisions.

5.2. Real world video streaming tests

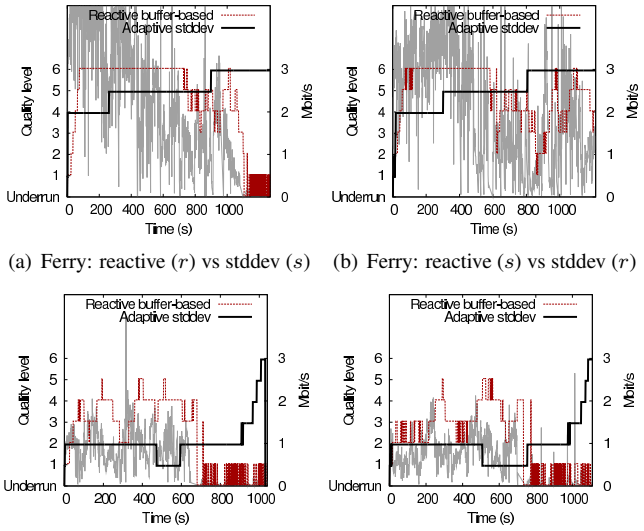
In the previous section, we used simulations to show the effectiveness of the prediction model, and in this section, we present the results from streaming sessions in the real environment for our two tested commute paths using the ferry and the metro as shown in figures 2(a) and 2(c). Since there are many factors influencing the real-world results, we present the results from two experiments for each route in figure 8. The graphs on the left (figures 8(a) and 8(c)) show the results with the reactive algorithm, and adaptive stddev is simulated on the same data and plotted in the same figure[4]. The graphs on the right show the same comparison, but we swapped the real-life and the simulated algorithm for the test. We see that the adaptive stddev algorithm provides a better viewing experience, having far fewer buffer underruns and more stable quality, confirming our simulated results.

6. CONCLUSION

In an effort to maintain stable video quality and reduce the number of playout interruptions due to fluctuating bandwidth and network outages, we have proposed using a GPS-based bandwidth-lookup service. Using video bitrate planning algorithms that make use of bandwidth predictions, our experimental results show that the video quality becomes more stable and that buffer underruns can often be avoided.

There are still several unaddressed challenges like for example GPS-power consumption, algorithm efficency and movement prediction. Ongoing and future work include finding better and more efficient prediction algorithms, and extending the commuting scenarios to forms of transportation

[4]Even though one is real and one is simulated, the plots are comparable since we showed in section 5.1 that the simulations and the real-world tests give the same results when using the same bandwidth logs.



(a) Ferry: reactive (r) vs stddev (s) (b) Ferry: reactive (s) vs stddev (r)

(c) Metro: reactive (r) vs stddev (s) (d) Metro: reactive (s) vs stddev (r)

Fig. 8. Comparing the reactive buffer-based and the prediction-based adaptive stddev algorithms in real-world experiments (r = real world, s = simulation).

that have less predictable movement patterns, such as cars. This requires algorithms that can handle frequent branches in the travelling routes. Furthermore, bandwidth sharing with other clients means that there may be time dependent variations in the bandwidth measurements. Our next version of the lookup service will take these aspects into account.

7. REFERENCES

[1] CURCIO, I. D. D., VADAKITAL, V. K. M., AND HANNUKSELA, M. M. Geo-predictive real-time media delivery in mobile environment. In *Proc. of MoViD - ACM MM workshops* (Oct. 2010), pp. 3–8.

[2] DIAZ-ZAYAS, A., MERINO, P., PANIZO, L., AND RECIO, A. M. Evaluating video streaming over GPRS/UMTS networks: A practical case. In *Proc. of IEEE VTC Spring* (Apr. 2007), pp. 624–628.

[3] GOYAL, V. K. Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine 18*, 5 (September 2001), 74–93.

[4] HUANG, J., KRASIC, C., WALPOLE, J., AND FENG, W. Adaptive live video streaming by priority drop. In *Proc. of IEEE AVSS* (2003), pp. 342–347.

[5] JOHANSEN, D., JOHANSEN, H., AARFLOT, T., HURLEY, J., KVALNES, Å., GURRIN, C., SAV, S., OLSTAD, B., AABERG, E., ENDESTAD, T., RIISER, H., GRIWODZ, C., AND HALVORSEN, P. DAVVI: A prototype for the next generation multimedia entertainment platform. In *Proc. of ACM MM* (Oct. 2009), pp. 989–990.

[6] LIVA, G., DIAZ, N. R., SCALISE, S., MATUZ, B., NIEBLA, C. P., RYU, J.-G., SHIN, M.-S., AND LEE, H.-J. Gap filler architectures for seamless DVB-S2/RCS provision in the railway environment. In *Proc. of IEEE VTC Spring* (May 2008), pp. 2996–3000.

[7] MOVE NETWORKS. Internet television: Challenges and opportunities. Tech. rep., Move Networks, Inc., November 2008.

[8] NETVIEW TECHNOLOGY. `http://www.netview.no/index.php?page=downloader`, 2010.

[9] NI, P., EICHHORN, A., GRIWODZ, C., AND HALVORSEN, P. Fine-grained scalable streaming from coarse-grained videos. In *Proc. of ACM NOSSDAV* (2009), pp. 103–108.

[10] PANTOS, R., BATSON, J., BIDERMAN, D., MAY, B., AND TSENG, A. HTTP live streaming. `http://tools.ietf.org/html/draft-pantos-http-live-streaming-04`, 2010.

[11] RIISER, H., HALVORSEN, P., GRIWODZ, C., AND HESTNES, B. Performance measurements and evaluation of video streaming in HSDPA networks with 16QAM modulation. In *Proc. of IEEE ICME* (June 2008), pp. 489–492.

[12] SCHIERL, T., DE LA FUENTE, Y. S., GLOBISCH, R., HELLGE, C., AND WIEGAND, T. Priority-based media delivery using SVC with RTP and HTTP streaming. *Multimedia Tools and Applications (MTAP)* (2010), 1–20.

[13] SCHWARZ, H., MARPE, D., AND WIEGAND, T. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology 17*, 9 (September 2007), 1103–1129.

[14] WAC, K., VAN HALTEREN, A., AND KONSTANTAS, D. Qos-predictions service: Infrastructural support for proactive qos- and context-aware mobile services (position paper). vol. 4278 of *Springer Lecture Notes in Computer Science*. 2006, pp. 1924–1933.

[15] ZAMBELLI, A. Smooth streaming technical overview. `http://learn.iis.net/page.aspx/626/smooth-streaming-technical-overview/`, 2009.

[16] ZINK, M., KÜNZEL, O., SCHMITT, J., AND STEINMETZ, R. Subjective impression of variations in layer encoded videos. In *Proc. of IWQoS* (2003), pp. 137–154.