

# Video Streaming Using a Location-based Bandwidth-Lookup Service for Bitrate Planning

HAAKON RIISER, TORE ENDESTAD, PAUL VIGMOSTAD,  
Netview Technology AS, Norway

CARSTEN GRIWODZ and PÅL HALVORSEN,  
Department of Informatics, University of Oslo, Norway & Simula Research Laboratory, Norway

---

A lot of people around the world commute using public transportation and would like to spend this time viewing streamed video content such as news or sports updates. However, mobile wireless networks typically suffer from severe bandwidth fluctuations, and the networks are often completely unresponsive for several seconds, sometimes minutes. Today, there are several ways of adapting the video bitrate and thus the video quality to such fluctuations, e.g., using scalable video codecs or segmented adaptive HTTP streaming that switches between non-scalable video streams encoded in different bitrates. Still, for a better long-term video playout experience that avoids disruptions and frequent quality changes while using existing video adaptation technology, it is desirable to perform bandwidth prediction and planned quality adaptation.

This paper describes a video streaming system for receivers equipped with a GPS. A receiver's download rate is constantly monitored, and periodically reported back to a central database along with associated GPS positional data. Thus, based on the current location, a streaming device can use a GPS-based bandwidth-lookup service in order to better predict the near-future bandwidth availability and create a schedule for the video playout that takes likely future availability into account. To create a prototype and perform initial tests, we conducted several field trials while commuting using public transportation. We show how our database has been used to predict bandwidth fluctuations and network outages, and how this information helps maintain uninterrupted playback with less compromise on video quality than possible without prediction.

Categories and Subject Descriptors: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Video; C.2.2 [Computer-Communication Networks]: Network Protocols—Applications

General Terms: Experimentation, Measurement, Performance

Additional Key Words and Phrases: adaptive streaming, bandwidth prediction, bitrate planning, mobile internet, wireless, fluctuating bandwidth, GPS

---

## 1. INTRODUCTION

Mobile wireless streaming is an increasingly popular service. All over the world, commuters on buses, trains, boats, etc. are watching videos on mobile devices such as smart phones, tablet PCs, laptops, etc. To support a true on-demand service, the video can not be downloaded in advance, but needs to be

---

This work has been performed in the context of the *HyStream* project (project number 176847) and the *iAD* centre for Research-based Innovation (project number 174867) – both funded by Norwegian Research Council.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies show this notice on the first page or initial screen of a display along with the full citation. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers, to redistribute to lists, or to use any component of this work in other works requires prior specific permission and/or a fee. Permissions may be requested from Publications Dept., ACM, Inc., 2 Penn Plaza, Suite 701, New York, NY 10121-0701 USA, fax +1 (212) 869-0481, or [permissions@acm.org](mailto:permissions@acm.org).

© 2011 ACM 1551-6857/2011/-ART0 \$10.00

DOI 10.1145/0000000.0000000 <http://doi.acm.org/10.1145/0000000.0000000>

streamed to the mobile device. Unfortunately, it is well known that mobile wireless networks typically suffer from fluctuating bandwidth [Riiser et al. 2008] and that the available bandwidth varies as the device is brought in and out of areas with network coverage.

Traditional media streaming services did not support automatic quality and bitrate adaptation, and could not easily compensate for significant variations in bandwidth. However, developments in adaptive streaming have made it possible to change bitrates as the network bandwidth changes, e.g., using scalable codecs, like Multiple Description Coding (MDC) [Goyal 2001], Scalable Video Coding (SVC) [Schwarz et al. 2007] and scalable MPEG (SPEG) [Krasic et al. 2003], or the popular adaptive HTTP streaming approaches from companies such as Move Networks [Move Networks 2008], Microsoft [Zambelli 2009], Apple [Pantos et al. 2010], and Netview Technology [Netview Technology 2010]. 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, and it is up to the applications to determine how to use the adaptation mechanisms. However, choosing when and how much data to retrieve, and which bitrates to use, are all 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 [Ni et al. 2009; Zink et al. 2003]. Furthermore, when using mobile Internet devices, it is a common occurrence to have the connection go down for significant periods of time, e.g., when entering a tunnel or at sea, far from the nearest base station.

At best, such losses of connectivity result in frequently changing, low quality video presentations, but video playout hiccups due to buffer underrun are also common. Either way, the effect is harmful to the users' quality of experience (QoE). These scenarios would greatly benefit from capacity planning which could be used to pre-buffer and smoothen the video quality over a longer period. However, if nothing is known about the likelihood of bandwidth fluctuations, the only safe 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. Ideally, the receiver would know in advance precisely how the network behaves over the course the streaming session, regardless of how the user moves around with the streaming device.

In this paper, we show how video receivers equipped with Global Positioning System (GPS) devices can be used to build a database that enables bandwidth prediction through geographical location and past network conditions. Even few samples per position are sufficient. Using a *GPS-based bandwidth-lookup service*, the application can predict future fluctuations in bandwidth for a trip along a planned route and present a smoother, better quality video stream to the users<sup>1</sup>. We present several real field trials where we have measured available bandwidth along popular commute routes in Oslo (Norway). The data sets from these measurements are used to build a prototype of our GPS-based bandwidth-lookup service, which again is used for bandwidth planning in a video streaming scenario, enabling smarter pre-buffering and quality adaptation. Finally, we have developed an adaptive bandwidth prediction algorithm that combines a reactive buffer-based algorithm with a simple prediction model. Using real-life test data, we first simulated video streaming using several algorithms, and compared the results with both the ideal case (an omniscient algorithm with complete a priori knowledge) and the classical case (no knowledge of the receiver's geographical location, using only currently buffered data and observed bandwidth when scheduling quality). We implemented the best performing algo-

<sup>1</sup>Non-GPS clients can also use this information for planning trips using public transport, although they cannot adapt to unexpected situations. This kind of use is beyond the scope of this paper.

rithm in Netview's adaptive media player [Netview Technology 2010], and did real video streaming sessions along different commute routes to confirm that our method works just as well in the real world.

The real world tests show that our bandwidth prediction system gives significant improvements in perceived video quality. Compared to the reactive algorithm, the predictive algorithm has less unused video in the buffer at the end of the trip, it experiences fewer playout interruptions and less changes in video quality.

## 2. LOCATION-BASED BANDWIDTH LOOKUP SERVICE

Geographic location has a great influence on network conditions experienced by mobile devices. For example, a receiver can be located on open ground near a base station, in which case the signal should be excellent, or in a tunnel or mountainous area far from a base station. In a mobile streaming scenario, it would be beneficial to predict future network conditions based on the current and future geographical location. When a user commutes, it is appropriate to assume a predictable sequence of locations. Reliable public transport additionally enables fairly good predictions of the times when locations are visited.

The basic idea with the location-based bandwidth lookup service is to monitor a streaming session's download rate and geographical location, and to upload this data to a central service. This service enables all users to query for predicted available bandwidth for given locations based on previous observations, even without requiring that uploaders' identities are stored alongside the position information they provided. Today, many new mobile phones and other mobile devices are capable of this. A receiver with a GPS can log network conditions along with GPS data while viewing video. By uploading them periodically, the viewers can help build a database that can provide an invaluable service to future users. While moving along a particular path (e.g., a popular commute route) the streaming application queries the bandwidth database for average bandwidths for every position along this path. These numbers are then used to predict and prepare for connectivity gaps, and to smoothen out the video quality along a route with highly fluctuating bandwidth. In a public transport scenario, a full prediction and a corresponding schedule can be calculated a priori using the GPS-based lookup service to verify the timely progress and validate the prediction.

Making efficient use of such a data set is possible because reception quality in mobile networks is often explained by geographical location. Combining this with the fact that many travel routes are highly predictable (especially when considering commuters using public transportation), the value of such a database becomes clear. It greatly empowers algorithms for pre-buffering and early quality adaptation. The main motivation for using these algorithms is to provide a more stable video viewing experience: planned quality adaptation can lead to more gentle quality switching, and bandwidth drops can be canceled out by pre-buffering during high-bandwidth situations.

## 3. BANDWIDTH MONITORING

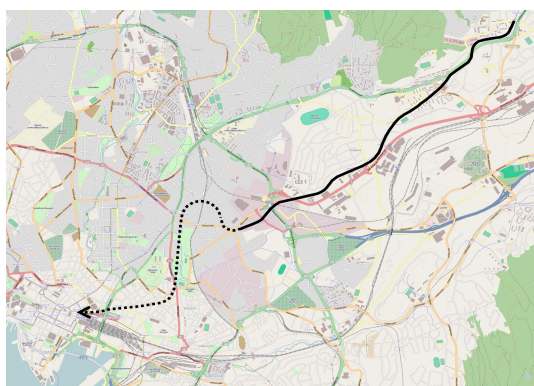
To build a data set for our prototype of the location-based bandwidth-lookup service, we performed field trials along four popular commute routes to downtown Oslo, Norway. These routes use different types of public transportation: metro, bus, ferry and tram, where the respective paths are highlighted in figures 1(a), 2(a), 4(a) and 5(a). The general idea of 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 (Universal Mobile Telecommunications System (UMTS)) in Oslo. We have earlier reported significant fluctuations in network conditions with this kind of network [Riiser et al. 2008], making it a good test case. To perform the field experiments in this study, we used laptops with GPS (Haicom HI-204III

USB GPS) and 3G mobile Internet devices (Huawei Model E1752 HSPA USB stick) as mobile video streaming devices.

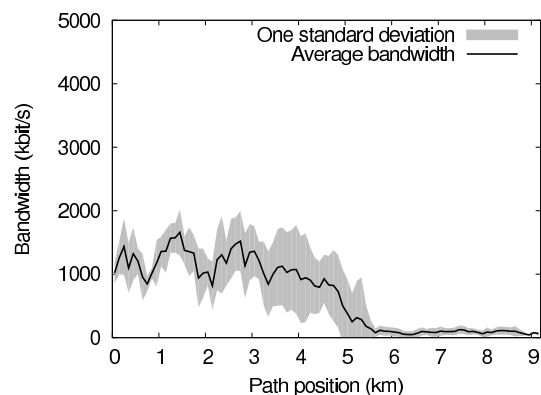
The bandwidth measurements were conducted by performing continuous HTTP downloads in an adaptive segmented HTTP streaming system [Johansen et al. 2009]. In our test scenario, the receiver has all required information available, including the user's actual location, intended destination and route, and expected bandwidths. In our measurements, we counted the number of bytes received every second and reported this along with GPS coordinates to a remote database. We conducted bandwidth measurements several times for every route where the plots show the average bandwidths as a function of the path position (as traveled distance from the start) with highlighted standard deviation.

### 3.1 Metro railway

A popular means of commuting in Oslo is the metro. This is an electric passenger railway, all of whose lines pass through underground tunnels in downtown Oslo. The underground part of our tested metro commute route is shown with the dotted line in figure 1(a). When the train was underground, we experienced a very poor signal. This means that we could benefit from choosing a low video bitrate before entering the tunnels, pre-buffering enough video to last through the tunnels.



(a) Metro commute path (the dotted parts indicate that the train is underground).



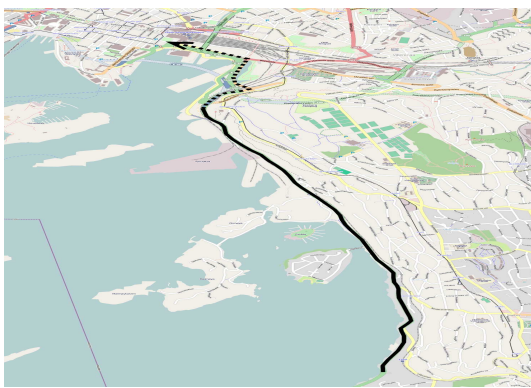
(b) Measured bandwidth along the metro commute path. One second average bandwidth.

Fig. 1: Oslo metro between Kalbakken and Jernbanetorget.

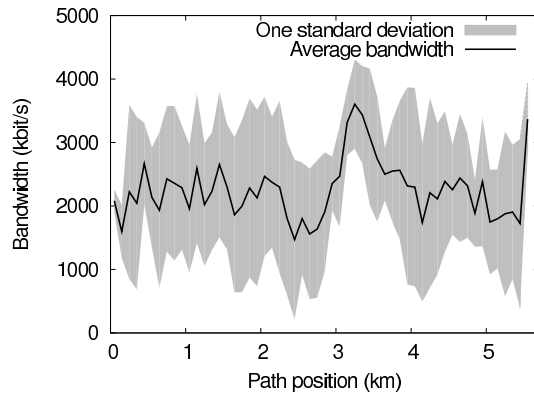
Figure 1(b) shows the measured bandwidth along the metro path. All the measurements show the same trend, and the signal and bandwidth availability are predictable with only minor variations. The experienced bandwidth is typically around 1 Mbit/s when the train is not underground. This means that it should be possible to stream videos at very high quality for most mobile devices (see table I for the bitrates available in our adaptive streaming service). However, when entering the tunnels after approximately 5.5 kilometers, both the Internet connection and the GPS signal are essentially lost (in periods without a GPS signal, the position is estimated based on the metro time table). With respect to the total commuting time on this route, about 55 % is spent outside the tunnels. This means that, with sufficient information about bandwidth availability when entering the tunnels, a video streaming system with a smart bitrate planning algorithm can stream less than 1 Mbit/s before the tunnel, and use the remaining bandwidth to prepare for the low bandwidth in the tunnel.

### 3.2 Bus

Figures 2(a) and 2(b) show a bus path going into Oslo and the corresponding bandwidth measurements. The average values in figure 2(b) vary greatly, but the measurements show that a streaming session of about 1.5 Mbit/s should normally be possible. However, the example also shows one of the challenges of establishing this kind of a lookup service. The bus route has a steep hill on the east side, which prevents the reception of signals from eastern GPS satellites. The Oslofjord is in the west, leaving few possible sites for UMTS towers on that side. Consequently, both the GPS and the UMTS signals vary a lot, which again creates uncertainty in estimating where an achieved bandwidth is measured. This helps explain the wide standard deviation “belt” in figure 2(b). One reason for this is that our GPS device was unable to get a good signal in some experiments, but we did generally not observe many connection losses (only one single serious outage, typically lasting 20–30 seconds, occurring at a predictable location every time). To illustrate this, figure 3(a) shows the paths reported by the GPS devices on three trips along the same road. The deviation is probably caused by weak GPS signals and

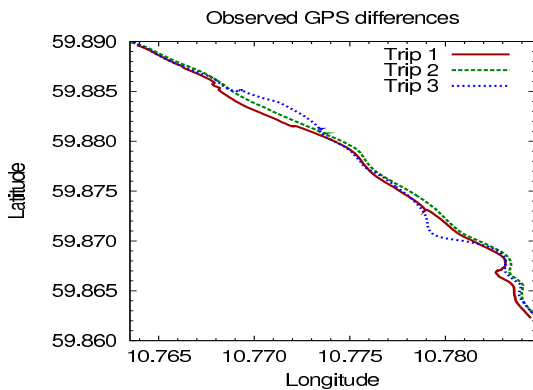


(a) Bus commute path (dotted parts excluded from the test because of road construction).



(b) Measured bandwidth along the bus commute path. One second average bandwidth.

Fig. 2: Oslo bus along Mosseveien, between Ljan and Oslo Central Station.



(a) Three different measurements along parts of the route.



(b) Difference between trip 2 and 3 (in figure 3(a)) at 59.8708 North on the map.

Fig. 3: Observed GPS differences on the bus route.

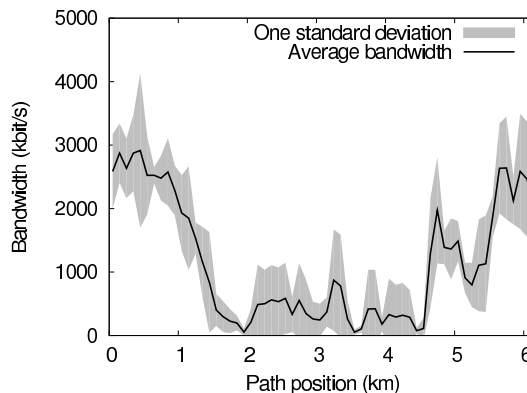
a GPS device which responds by returning extrapolated coordinates; for example, at latitude 59.8708 North (signed decimal degrees), trip 2 and 3 have longitude values of 10.7793 East and 10.7809 East, respectively. This difference translates to about 90 meters, i.e., driving the bus in the water (see figure 3(b)). Nevertheless, a lookup service can still be provided, and the uncertainty can be reduced with a sufficient number of measurements.

### 3.3 Ferry

Our third test scenario is travelling by boat (figure 4(a)). Most commuters from the Nesodden peninsula travel by ferry to Oslo, as traveling by car requires a large detour. With lots of space on board and tables for PCs and devices with large screens, this means of transportation is the one that is best suited for mobile video viewing of high resolution content. Signal conditions far from land can sometimes be problematic. Figure 4(b) shows that the available bandwidth varies quite a lot, but the signal is 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).



(a) Ferry commute path.



(b) Measured bandwidth along the ferry commute path. One second average bandwidth.

Fig. 4: Oslo ferry route between Nesodden and Aker Brygge.

### 3.4 Tram

The fourth way of commuting in Oslo using public transportation is by tram. Figure 5(a) shows the tested tram path, whose tracks are parallel to but high above the bus route. Figure 5(b) presents the measured bandwidth. Along the whole path, we have found acceptable but fluctuating bandwidth. In the first part of the route, we observed a very predictable bandwidth across the different measurements. At a long downslope towards the end of the trip, the measurements vary more.

## 4. VIDEO STREAMING WITH A GPS-BASED BANDWIDTH-LOOKUP SERVICE

The architecture of our GPS-based bandwidth-lookup service is illustrated in figure 6, and during a streaming session, the following three steps are performed:

- (1) The receiver chooses a destination and an appropriate route. It sends the route to the bandwidth database, providing the latitude and longitude for every 100 meters along the path. Using 32-bits

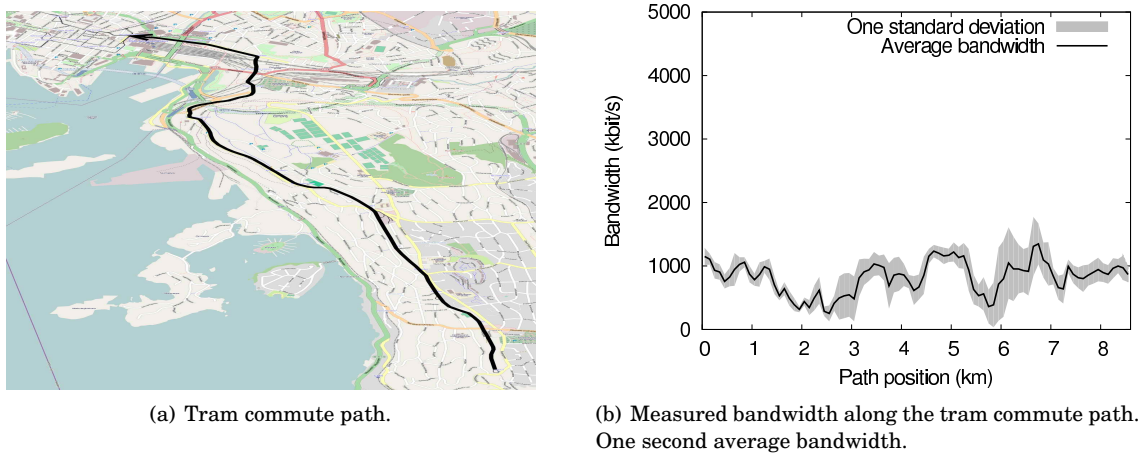


Fig. 5: Oslo tram route between Ljabru and Oslo Central Station.

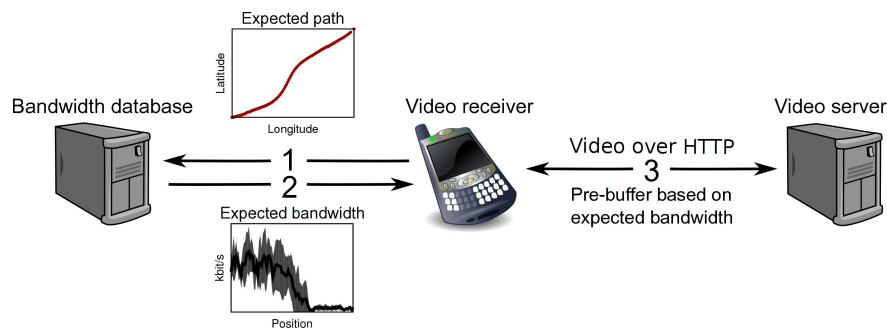


Fig. 6: System architecture.

per coordinate, the path description requires only eight bytes per 100 meters, so it is straightforward to send the entire path. In an actual product based on this approach, each receiver would record its routes, and path predictions would be based on the receiver’s travel history.

- (2) Once the bandwidth database receives the path description, it returns a sequence of bandwidth samples for each point listed in the path description. Each bandwidth sample includes both the average bandwidth and the standard deviation. It is sufficient to use 16-bit integers for both numbers, so the historic bandwidth data requires only four bytes per 100 meters.
- (3) During operation, the receiver combines the measured bandwidth for every 100 meters traveled with historic data of estimated traveling time. It can then calculate the estimated number of bytes that it can download during the remaining time of the trip. Equipped with bandwidth prediction, path and speed prediction, and knowing the average bitrate of each quality level, the receiver can plan which quality levels to use, and start downloading. For every downloaded segment, the receiver repeats this calculation, taking the number of bytes already in the buffer into account.

The goal is to have an empty buffer at the end of the trip, while only going  $up^2$  in quality, while at the same time minimizing rapid fluctuations in quality and avoiding buffer underrun. When the receiver detects that it deviates from the predicted route or timeline, it makes a new path prediction, and fetches a new set of bandwidth predictions.

Using this approach, it is possible to predict the amount of data that can be downloaded in the future, and thus, an outage in the network signal does not need to cause an interruption: the effects of short losses of network connection as well as predicted long connection losses are “smoothed” out over time.

#### 4.1 Test system

When testing our proposed solution in an adaptive segmented HTTP streaming scenario, we used an enhanced version of Netview’s proprietary media player and media downloader. This system operates in a similar way to the other modern HTTP streaming systems from Move, Apple and Microsoft, and it supports several adaptive streaming formats. For this test, we chose to use a modified version of Apple’s Live HTTP streaming format. We used the same playlist format, but to save space for the lower bitrate quality levels, we used Netview’s own multiplexing container instead of the significantly less efficient MPEG-2 Transport Stream format typically used with Apple Live HTTP [Riiser et al. 2010].

For content, we used European football (soccer) matches encoded in six quality levels (see table I), all using a fixed segment length of two seconds and 25 frames per second. Quality levels range from low-resolution/low-bitrate video for old mobile phones to HD quality for laptops with big displays. We plan long-term and do not need to adapt quickly. Thus, we do not have need for H.264/SVC, which is not available on most mobile devices today anyway. Akamai [2010] recommends that video for 3G networks should be encoded at 250 kbit/s for low quality and 450 kbit/s for high quality, i.e., very close to our chosen quality levels (the reason for also having bitrates far above 450 kbit/s is that our configuration does not target *just* small portable devices, but also laptops and tablet PCs that benefit from HD resolutions).

Figure 7 shows the bitrate distribution for all the six quality levels, and one can see that, while the bitrate is not constant over each two second segment, the deviation from the average is small, so using the average bitrate for predictions should not be a problem.

Quality level	kbit/s
1	250
2	500
3	750
4	1000
5	1500
6	3000

Table I. : Video quality levels

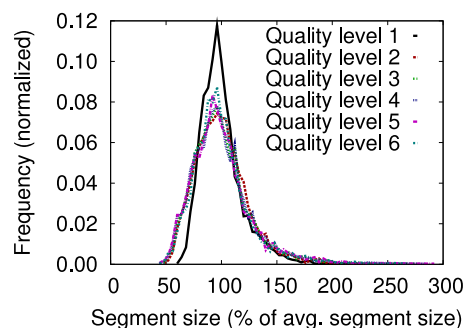


Fig. 7: Video bitrate distribution

<sup>2</sup>We aim for an increasing quality, because 1) users strongly tend to evaluate the quality better compared to starting high and lowering the quality towards the end [Zink et al. 2003], 2) starting low reduces the startup time [Johansen et al. 2009] and 3) being careful in the beginning helps build a buffer faster, making the session less prone to buffer underruns.



Note that when proving the concept of our lookup service in our initial tests, and when demonstrating the effect of bandwidth prediction, we did not implement path prediction. We performed this operation by hand, manually identifying which of our chosen commute paths to use before starting the streaming session. On-the-fly path prediction and management of general, more random movement (e.g., a car with passengers that stream video) are future enhancements.

## 4.2 Algorithms

The described location-based bandwidth lookup service calculates and predicts the available bandwidth along a path. Obviously, reported entries in the database do not have identical position and bandwidth measurements, and recording time as well as traffic patterns vary as well. Considering this, it is up to the application to determine how the predicted bandwidth should be used. In our study, we have tested and compared two different algorithms for quality adaptation. We also include in the comparison an omniscient algorithm that shows the “optimal” result. One of our algorithms represents current systems with a reactive approach (no prediction), and the predictive algorithm uses our database with GPS-tagged historic bandwidth measurements.

### A buffer-based reactive algorithm

The basic idea of the buffer-based reactive algorithm is to select the video bitrate (and thus quality) based on the number of seconds of video that are preloaded in the buffer. Thus, when the buffer reaches a certain size for a given level, the system is allowed to increase the quality. Similarly, when draining the buffer, the selected quality is reduced if the buffer shrinks below the threshold.

Such an algorithm works in most cases, but the quality oscillations follow the bandwidth availability oscillations, and during network outages, buffer underruns and playout interruptions do occur. The used reactive algorithm therefore makes a trades off average quality for better protection against oscillations and playout interruptions. We put a cap on the quality level corresponding to the resource availability, i.e., avoid selecting a higher quality level than supported by the current available bandwidth. Additionally, we scale the buffer fullness thresholds for changing quality according to the bitrate differences between quality layers, and to further avoid frequent oscillations in quality, the threshold for increasing the quality level is higher than when lowering the quality level.

### A history-based prediction algorithm

To make use of the location-based bandwidth lookup service, we use a history-based prediction algorithm that predicts future bandwidth using previously recorded bandwidth numbers. In this method, a path prediction is created where, for every 100 meters along the path, the system queries for a predicted bandwidth. The prediction is simply the average of historical (but recent) bandwidth numbers within 100 meters of this position. Each time a new video segment is to be downloaded, the algorithm takes the path, travel time and bandwidth predictions, current position and buffer fill level, and calculates the highest quality level that could be used for the rest of the trip without getting a buffer underrun anywhere.

In many cases, this algorithm performs very well. However, our experiments show that there are several factors that may influence the results in a real environment, such as accuracy of the GPS signal, competition for bandwidth, base station failure, unpredicted outages and traveling delays. In other words, there are situations where the prediction fails and buffer underruns do occur. There are several ways to alter the prediction when the bandwidth deviates from the average. We have earlier experimented with different algorithms for making predictions with configurable optimism, e.g., one was a complex Monte-Carlo method and another simply reduced the predictions based on bandwidth variance for each position. However, our experiments with self-correcting, purely predictive quality

selection algorithms were unsuccessful, and we found that the best solution was to add the reactive algorithm<sup>3</sup> as a “safety net”. This means that, for each segment to download, we also make a quality choice based on the reactive algorithm, compare it with the choice made using the prediction (described above), and choose the lowest of the two quality levels.

#### An omniscient prediction algorithm

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. It chooses bandwidths in the same way as described in the predictive algorithm above: For every segment, the maximum quality level that could be supported for the rest of the trip is used. Because the omniscient algorithm always has a perfect prediction, it never needs to reduce quality, and consumes its buffers completely.

### 4.3 Simulation

Performing real streaming experiments in a live, mobile environment for all configurations of the system is very time consuming, and to find the best algorithms to use and a suitable configuration, we first performed a set of simulations. In these experiments, we used the test content described in table I, and for a chosen bandwidth log (from our initial real measurements of network bandwidth), we simulated a media player that 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.

Our first task was to find a suitable configuration of both the reactive and the predictive algorithms with respect to handling variations in the network availability. Due to space limitations, the more detailed investigations are described in appendix A and appendix B, respectively, where we discuss and show experimentally the algorithms’ behavior when changing a particular parameter. In summary, the used *reactive* algorithm (see appendix A) applies: 1) a quality capping function limiting the selected quality level to available bandwidth where the bandwidth is calculated using a moving average with a weight of 10 % on the last sample (one bandwidth sample per second); 2) a step scaling function using a base step size of 10 seconds buffer length multiplied with the relative bandwidth requirement difference between the two quality layers, i.e., making the buffer fill level requirement higher for switching to higher layers; and 3) a method for avoiding rapid fluctuations in quality by having different thresholds for going up and down in quality (going up requires 20 % more data in the buffer than going down), and delaying quality upgrades by 20 seconds after a drop in quality. Furthermore, the *predictive* algorithm (see appendix B) applies: 1) the reactive algorithm as a “safety net”, i.e., overriding the predictive algorithm when the buffer fill level is lower than what would be accepted by the reactive algorithm; 2) a function to take variable-duration tips into account, avoiding to increase the quality level when less than 15 % of the trip remains, using the available bandwidth to increase the safety margins instead; and 3) a video quality oscillation control preventing too frequent quality changes.

Having found useful configurations for the algorithms, we present in figure 8 comparisons of the reactive buffer algorithm, the predictive algorithm and the omniscient algorithm for four different bandwidth logs (one for each of our chosen commute routes). The first observation is that the traditional reactive algorithm follows the available bandwidth fairly well. The video quality drops when a bandwidth drop occurs, but naturally, the quality drop in video that is played out occurs later because of the amount of prefetched data in the buffer. Additionally, the results show that the capping, the

<sup>3</sup>Bandwidth capping is not enabled in combination with the predictive algorithm, because it interferes with the long term planning required to smooth out variations in bandwidth.

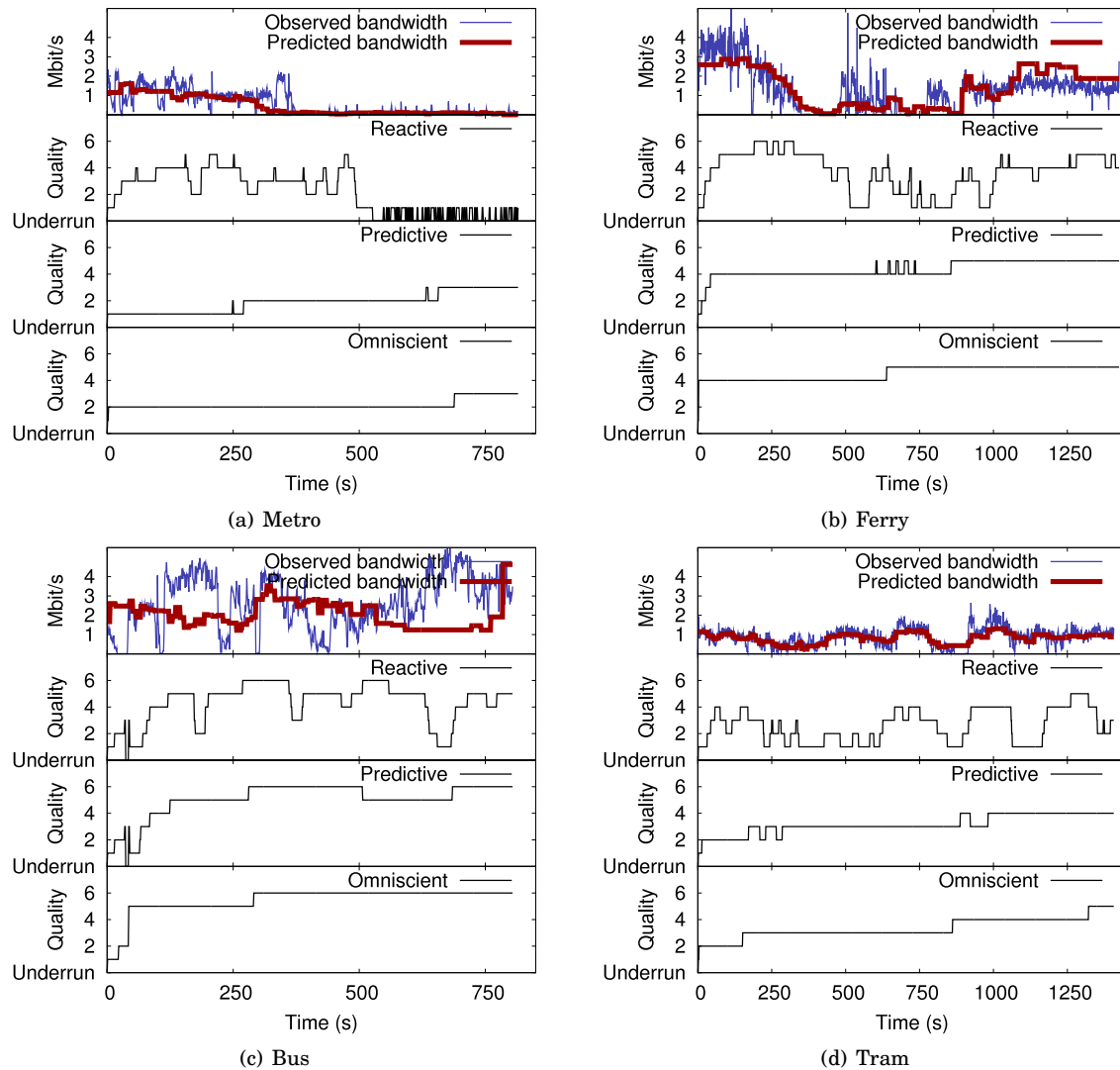


Fig. 8: Video quality levels for the reactive, prediction and omniscient algorithms.

step size scaling and the different up/down limits are able to deal with small network outages, but the quality still oscillates heavily. For longer times without network connection like on the metro, the reactive algorithm fails to provide a continuous video playback. We see that the predictive algorithm better handles network variations and outages. The video quality is smoothed out as the algorithm can use the predicted bandwidth in the calculations, and even large outages (like in a tunnel or at mid-sea) can be foreseen and managed. For example, in the metro tunnel, the reactive algorithm drains the buffer quickly, and the video playout stops approximately 300 seconds before arrival. The predictive algorithm calculates the average possible quality in advance, and by sacrificing quality at the beginning for prefetching bandwidth, the video playout lasts for the whole 820 seconds of the trip (with half

the time in a tunnel). From the results, we observe that the predictive system performs close to the omniscient algorithm.

Furthermore, for these tests, we present the buffer fullness in figure 9. In the plots, we can see 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. They do not stop downloading video until the player is shut down, and the playout position may be 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. Nevertheless, we observe again that the predictive algorithm follows the omniscient algorithm better than the reactive algorithm.

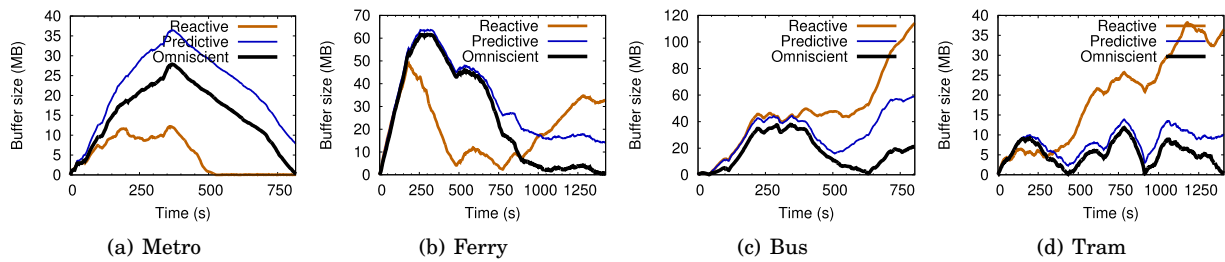


Fig. 9: Buffer-fullness for the reactive, prediction and omniscient algorithms

Finally, to evaluate the accuracy of our simulations, we first ran a real streaming session and then a simulation on the bandwidth log recorded in the real session. The results with respect to the quality level as a function of time are shown in figure 10. We can observe that the real and the simulated systems make almost exactly the same decisions.

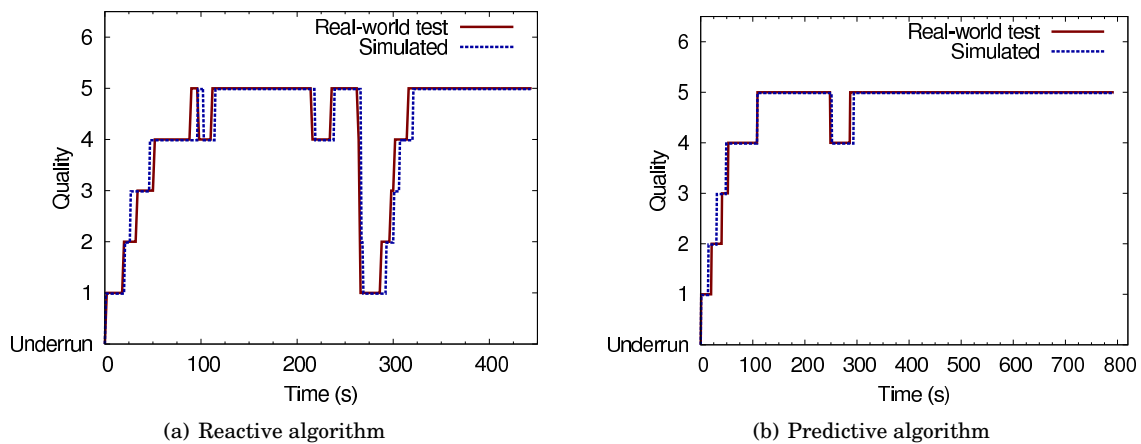


Fig. 10: Testing the accuracy of the simulations by comparing real-world tests of algorithms with the simulation (using the bandwidth measured in the real test to make them comparable)

#### 4.4 Real world video streaming tests

In the previous section, we used simulations to show the effectiveness of the prediction model. In this section, we show the results from real streaming sessions in the real environment for our tested commute paths using the bus, the metro, the ferry and the tram as shown in figures 1(a), 2(a), 4(a) and 5(a). In figure 12, we present the results where the graphs on the left show the result using a traditional reactive buffer based algorithm, while the graphs on the right show the results using the GPS-based bandwidth lookup service with the predictive algorithm. Additionally, in each figure, we have also plotted the results from a simulation over the same bandwidth trace captured during the real test using the other algorithm, i.e., the results should be directly comparable since we showed in the previous section that the real-world results directly map to the simulated results.

The plots in figure 12 show that the real-world tests confirm the simulation results. Using the bandwidth lookup service with the predictive algorithm, a user will experience far fewer buffer underruns and more stable quality. However, we also experienced that there are conditions that prevent error-free predictions. For example, we experienced buffer underruns shown in figure 11(a), because the metro was delayed (more than the 15 %-remaining time function could handle) in an underground station near the end of the trip. Because such delays occur very rarely, we failed to predict the duration of the trip, causing a buffer underrun without any way to recover. Another example is shown in figure 11(b), where the signal was almost completely gone for several minutes, at a location where previous measurements found acceptable network conditions (see the differences between the observed and the predicted bandwidth). Nevertheless, the real-world experiments show that the proposed system is well suited for video streaming in mobile, bandwidth fluctuating scenarios.

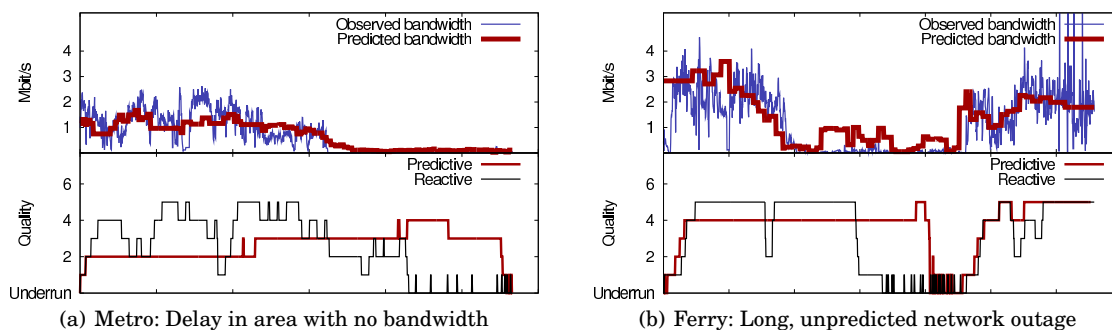


Fig. 11: Examples of failed predictions

## 5. DISCUSSION AND OPEN ISSUES

Our prototype proves that a GPS-based bandwidth lookup service can improve the quality of streaming sessions in mobile scenarios. However, there are still many parts of the system that need further improvements and open issues that must be solved before such a service is made available to the public.

For example, guaranteeing the consistency of the database of measurements is difficult. As shown in section 3.2, signals may be lost, positions may be inaccurate and the measurements may be corrupted. In our prototype, we have not filtered measurements, but the lookup service should do this. Similarly, we do not differentiate between different times of day, but there may, for example, be more people

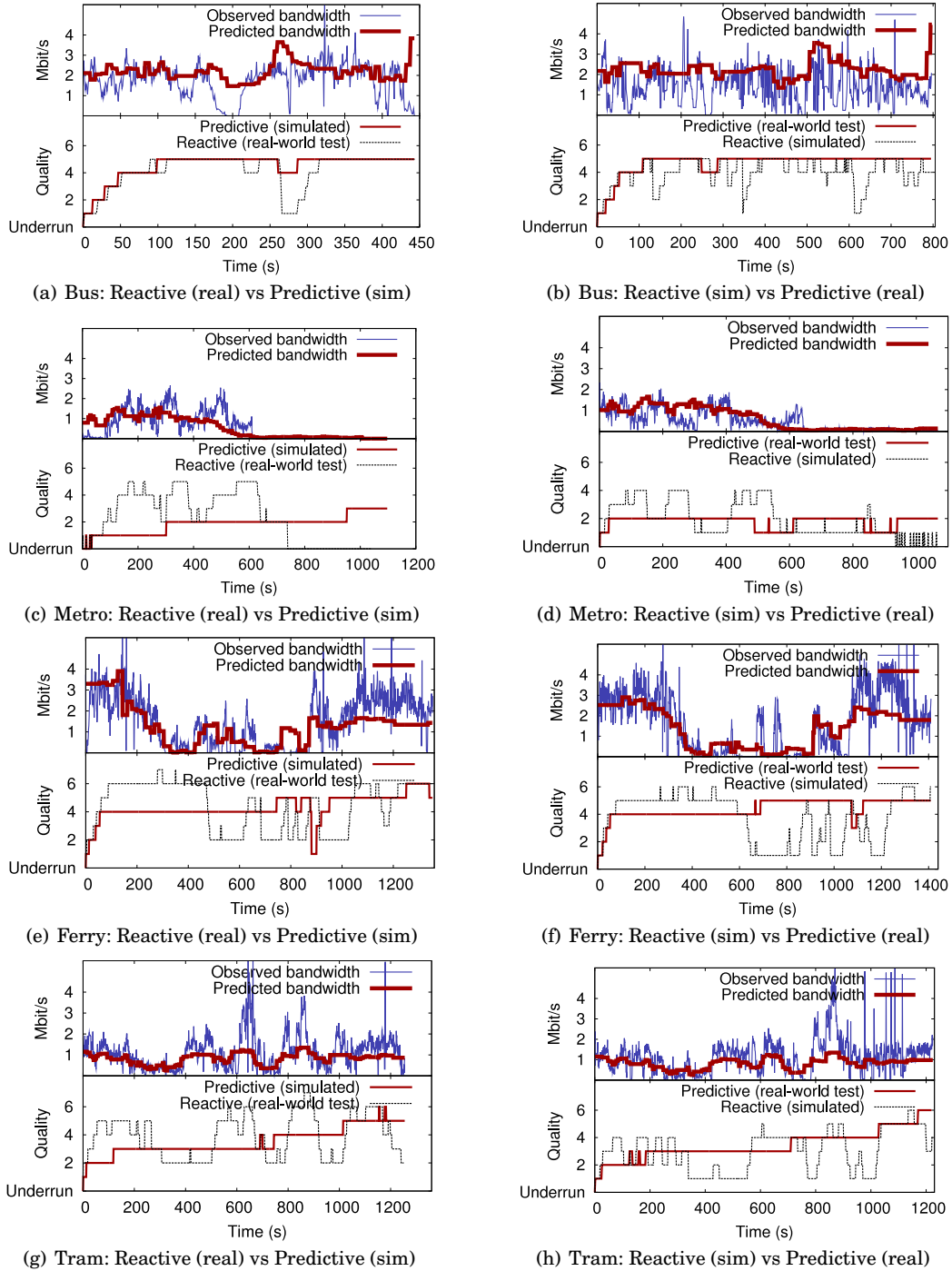


Fig. 12: Real-world experiments: the reactive buffer-based method vs. the predictive method.

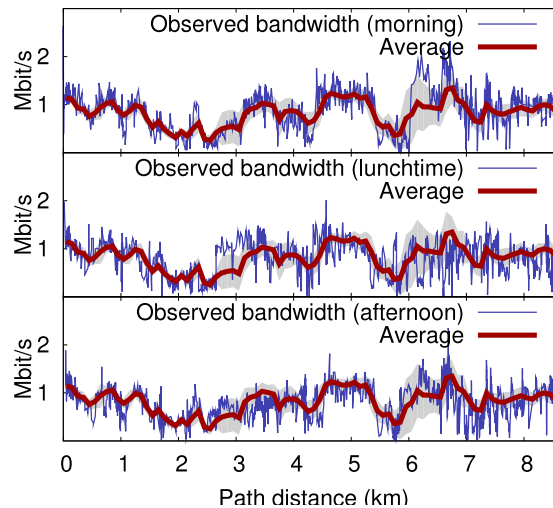


Fig. 13: Times of day differences.

competing for the bandwidth during rush hours or during regular week days where people more often commute. However, as shown in figure 13 where the measurements from the tram-route depicted in figure 5(b) are grouped by time, the time of day does not significantly influence the results<sup>4</sup>. For this initial study, however, we do therefore not differentiate the returned results according to time or date. Nevertheless, if future measurements indicate time differences, the lookup service can easily take time of day and day of week into account. Moreover, a related issue is that the current lookup prototype returns the predicted bandwidth, but the current competition for bandwidth (number of concurrent users) is not taken into account. Thus, the available bandwidth predicted by the lookup service might be shared transparently. This situation will, however, be improved by allowing the system to return different statistics according to day-of-week and time-of-day as we describe above, where such concurrent use over time is automatically reflected in the predicted bandwidth.

As the users move around both randomly and along predefined paths, at different speeds, etc., a question that arises is how far into the future a prediction should reach. In our experiments, we have assumed public transportation where the route and the approximate speed (timing) are known. However, the idea can be applied perfectly well to more individual means of transportation such as cars, whose direction and speed is unpredictable. Such scenarios are outside the scope of this paper, but they raise several challenges that should be addressed.

We have chosen to demonstrate the bitrate planning idea using adaptive HTTP streaming with multiple AVC versions of a segment as shown in table I. The reason is of course the current popularity of this streaming solution and the availability of Netview’s media player and the DAVVI system [Johansen et al. 2009] to perform live streaming tests. However, this also means that when a decision is made with respect to the video quality of a segment, one must either stick with the chosen quality for the duration of the segment or re-download the complete segment in another quality, possibly wasting bandwidth. This makes planning more vulnerable to “unpredicted” variations. Nevertheless, there are no restrictions on the lookup service that prevents it from being applied with other adaptive streaming

<sup>4</sup>This will probably change in the future when even more people use the streaming services, and we do have ongoing activities looking into these challenges.

solutions. For example, using a scalable codec like SVC and building up playout windows enhancing the video quality layer by layer until the resources are consumed (like PALS [Rejaie and Ortega 2003] or priority progress streaming [Huang et al. 2003]) would, with respect to the bit planning, give a simpler and possibly a more efficient bitrate planning solution (the inherent overhead of scalable codecs notwithstanding).

To improve the last-hop bottleneck, there are several possible approaches. One orthogonal approach is our activity in multi-link streaming [Kaspar et al. 2010; Evensen et al. 2010]. Here, the system transparently splits the TCP connection over multiple available (wireless) interfaces, and data segments are dynamically split up and transmitted on the different links in parallel to increase the available bandwidth. Such an approach could be combined with the proposed bitrate planner.

## 6. RELATED WORK

Video streaming has been a hot research topic for a couple of decades, and many video services are available today. A remaining challenge is to adapt video streaming to the unpredictable behavior of wireless networks like General Packet Radio Service (GPRS) and High-Speed Packet Access (HSPA). With mobile receivers in such networks, fluctuating network bandwidths strongly influence the video streaming service performance [Riiser et al. 2008; Diaz-Zayas et al. 2007], raising a need for bitrate adaptation to cope with temporary connection loss, high packet error rate and insufficient channel capacity, etc.

In order to reduce the effects of transmission bitrate variations or temporary link interruptions caused by, for example, changes in the channel conditions or the wireless interface, one important strategy has been to adapt video quality, and thus resource consumption, to resource availability. Currently, several commercial systems like Apple's HTTP Live Streaming [Pantos et al. 2010] and Microsoft's Smooth Streaming [Zambelli 2009] monitor the download speed of video segments and dynamically adapt to resource availability changes by switching between video segments coded in different qualities and bitrates. This is a highly popular approach, but a potential challenge is too frequent quality switches which may annoy the users [Zink et al. 2003; Ni et al. 2009], i.e., these systems use a traditional preloading approach, smoothening out the quality changes over a larger window.

Several similar preload approaches use (hierarchical, layered) scalable video codecs like scalable MPEG (SPEG) [Huang et al. 2003; Krasic et al. 2003], Multiple Description Coding (MDC) [Goyal 2001] and the Scalable Video Coding (SVC) extension to H.264 [Schwarz et al. 2007] to adapt to varying network conditions. For example, priority progress streaming [Huang et al. 2003] makes a prefetch window which is first filled with the scalable codec's base layer. If there is more time (and bandwidth) before the window should be played out and the next window started to be filled, quality enhancing layers are downloaded. Schierl et al. [2010] propose a priority-based media delivery for wireless 3GPP using SVC. A transmission scheduling algorithm prioritizes media data according to its importance in both RTP and HTTP-based streaming scenarios. This system prebuffers low-quality (but high priority) data of a given window length and increase the quality if the buffer is filled.

A cross-layer design to improve the H.264 video stream over IEEE wireless networks is proposed by Mai et al. [2010]. This system discovers the importance of packets containing AVC NAL-units according to their influence on picture quality in terms of PSNR and maps the video packets to appropriate access categories with shortest expected waiting time. The most important packets are sent first and late (low priority) packets can be dropped on the sender side.

The video playback may also be adapted to local resources like remaining battery power on mobile devices. A QoS adaptation model [Tamai et al. 2004] for mobile devices can be used to automatically adjust the video quality (and processing requirements) from the remaining battery power, desirable playback duration and the user's preference. A similar approach is presented by Hsu and Hefeeda



[2010], where a quality-power adaptation framework controlling the perceived video quality and the length of viewing time for both live and on-demand broadcast scenarios on battery-powered video receivers. The framework predicts the remaining battery life-time and adjust the amount of SVC-coded video data to receive and process.

There exist several approaches for delivering video streams to moving vehicles where the network is partitioned and there are temporal channel variations. For example, V3 [Guo et al. 2005] aims at streaming live video from traffic incidents by constructing a mobile ad-hoc network using a vehicle-to-vehicle approach, using the different cars to forward video data in a peer-to-peer manner. The cars are assumed equipped with GPS devices which enable the vehicles to track their location and trajectory.

There has also been performed experiments collecting network characteristics in such mobile scenarios. In Lee et al. [2010], traces of signal-to-noise ratios along with GPS coordinates from vehicles were captured to later objectively evaluate and compare different rate adaptation schemes in order to have the same test environment for all tests. Thus, these measurements were not used for predicting geographical bandwidth availability.

General QoS-prediction services and QoS-information systems have been suggested several times. Sun et al. [2005] describe an idea of using network resource awareness on the application level, but there is no QoS-prediction service, users only make predictions based on their own history. Wac et al. [2006a] suggest a general QoS-prediction service containing real end-to-end performance metrics that is both generated by and available to other users, and the feasibility of using such predictions based on geographical location, time and historical data was proved for a mobile health monitoring application [Wac et al. 2006b]. Similarly, Horsmanheimo et al. [2004] performed a trial that demonstrates the usefulness of location-aided planning and adaptive coverage systems for network planning and resource management of mobile networks. Their trial does not deal with adaptive media streaming, focusing instead on the accuracy of various location methods and evaluating the accuracy of signal level prediction as a function of the location. Furthermore, the data collected is intended for the network operators, not the users. Liva et al. [2008] propose a gap filler to provide bi-directional connectivity to train-based terminals. A propagation analysis using a wave-guide theory model derives a path loss law that can be used in a link budget calculation, i.e., calculating how the signal behaves to predict gaps based on parameters like propagating mode, the form of the tunnel, attenuation due to roughness of the walls and antenna insertion loss.

The listed techniques in this section have all shown the ability to adapt to oscillations in resource availability and to some degree deal with network disconnections. Some of the approaches, like adapting the video streaming/segment download and playout to local device resources (e.g., remaining battery power), are completely orthogonal to our approach. With respect to distributed resources, the challenge is to know how the availability fluctuates to increase the average video quality and reduce the number of buffer underruns. Existing systems often monitor the observed throughput, but foreseeing future degradations or even connectivity loss is hard. To better enable the streaming application to predict future bandwidth availability and then adapt the video quality, regardless of whether adaptation is performed using a scalable codec [Huang et al. 2003; Schierl et al. 2010], transcoding [Brandt and Wolf 2008] or multi-quality segments [Pantos et al. 2010; Zambelli 2009; Adobe 2010; Move Networks 2008], we propose to use a GPS-based bandwidth lookup service with a predictive video quality selection algorithm, i.e., similar to have been done by Wac et al. [2006b], but in a completely different application scenario.

Such an idea of a geo-predictive media delivery system has recently been developed in parallel to our work [Curcio et al. 2010]. The authors predict network outages and prebuffer video. However, they provide only simulation results, do not use adaptation, and do not consider route planning. In contrast, our results are based on a running prototype with months of real-life experiments (complemented by

simulations) that stream video to a mobile device whose user travels along fixed commuter routes. We adapt video quality to available bandwidth and present video bitrate planning algorithms that make use of the bandwidth lookup service.

## 7. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a GPS-based bandwidth-lookup service that is used for bitrate planning in a video streaming scenario. During a streaming session, the receiver's bandwidth is monitored and reported back along with the associated GPS positional data, i.e., the users themselves build up the database required for the service. Then, using such a service, the streaming application can predict future network outages and bandwidth fluctuations based on the previously measured bandwidths.

As a proof of concept, we implemented a prototype that was tested on popular commute routes using public transportation in Oslo, Norway. Along these routes, we measured the bandwidth and used the measurement to build a database for the lookup service. Then, we modified our existing adaptive HTTP streaming prototype [Johansen et al. 2009] to perform bitrate planning based on the information returned by the lookup service. Our experiments, both simulations and real-world tests, show that the experienced video quality can be smoothed and that the severe network outages can be handled.

There are several open issues and potential improvements. For example, we intend to use the information contained in our bandwidth lookup service to select from several available wireless networks, similar to the cognitive resource manager envisioned by Mähönen et al. [2006]. We are also currently working on topics such as time-of-day dependencies, measurement aging and arbitrary movement support. Nevertheless, our initial prototype has proven that such services can be built and that they can improve the streaming quality orthogonally to existing adaptation schemes.

## REFERENCES

- ADOBE. 2010. HTTP dynamic streaming on the Adobe Flash platform. [http://www.adobe.com/products/httpdynamicstreaming/pdfs/httpdynamicstreaming\\_wp\\_ue.pdf](http://www.adobe.com/products/httpdynamicstreaming/pdfs/httpdynamicstreaming_wp_ue.pdf).
- AKAMAI. 2010. Akamai HD for iPhone encoding best practices. [http://www.akamai.com/dl/whitepapers/Akamai\\_HDNetwork\\_Encoding\\_BP\\_iPhone\\_iPad.pdf](http://www.akamai.com/dl/whitepapers/Akamai_HDNetwork_Encoding_BP_iPhone_iPad.pdf).
- BRANDT, J. AND WOLF, L. 2008. Adaptive video streaming for mobile clients. In *Proc. of ACM NOSSDAV*. 113–114.
- CURCIO, I. D. D., VADAKITAL, V. K. M., AND HANNUKSELA, M. M. 2010. Geo-predictive real-time media delivery in mobile environment. In *Proc. of MoViD - ACM MM workshops*. 3–8.
- DIAZ-ZAYAS, A., MERINO, P., PANIZO, L., AND RECIO, A. M. 2007. Evaluating video streaming over GPRS/UMTS networks: A practical case. In *Proc. of IEEE VTC Spring*. 624–628.
- EVENSEN, K., KUPKA, T., KASPAR, D., HALVORSEN, P., AND GRIWODZ, C. 2010. Quality-adaptive scheduling for live streaming over multiple access networks. In *Proc. of ACM NOSSDAV*. 21–26.
- GOYAL, V. K. 2001. Multiple description coding: Compression meets the network. *IEEE Signal Processing Magazine* 18, 5, 74–93.
- GUO, M., AMMAR, M. H., AND ZEGURA, E. W. 2005. V3: A vehicle-to-vehicle live video streaming architecture. *Pervasive and Mobile Computing* 1, 4, 404–424.
- HORSMANHEIMO, S., JORMAKKA, H., AND LÄHTENMÄKI, J. 2004. Location-aided planning in mobile network—trial results. *Wireless Personal Communications* 30, 207–216.
- HSU, C.-H. AND HEFEEDA, M. 2010. Achieving viewing time scalability in mobile video streaming using scalable video coding. In *Proc. of ACM MMSys*. 111–122.
- HUANG, J., KRASIC, C., WALPOLE, J., AND FENG, W. 2003. Adaptive live video streaming by priority drop. In *Proc. of IEEE AVSS*. 342–347.
- JOHANSEN, D., JOHANSEN, H., AARFLOT, T., HURLEY, J., KVALNES, Å., GURRIN, C., SAV, S., OLSAD, B., AABERG, E., ENDESTAD, T., RIISER, H., GRIWODZ, C., AND HALVORSEN, P. 2009. DAVVI: A prototype for the next generation multimedia entertainment platform. In *Proc. of ACM MM*. 989–990.
- KASPAR, D., EVENSEN, K., ENGELSTAD, P. E., HANSEN, A. F., HALVORSEN, P., AND GRIWODZ, C. 2010. Enhancing video-on-demand playback over multiple heterogeneous access networks. In *Proc. of CCNC*. 47–51.

- KRASIC, C., WALPOLE, J., AND FENG, W.-C. 2003. Quality-adaptive media streaming by priority drop. In *Proc. of ACM NOSSDAV*. 112–121.
- LEE, K. C., NAVARRO, J. M., CHONG, T. Y., LEE, U., AND GERLA, M. 2010. Trace-based evaluation of rate adaptation schemes in vehicular environments. In *Proc. of IEEE VTC Spring*.
- LIVA, G., DIAZ, N. R., SCALISE, S., MATUZ, B., NIEBLA, C. P., RYU, J.-G., SHIN, M.-S., AND LEE, H.-J. 2008. Gap filler architectures for seamless DVB-S2/RCS provision in the railway environment. In *Proc. of IEEE VTC Spring*. 2996–3000.
- MÄHÖNEN, P., PETROVA, M., RIIHIJÄRVI, J., AND WELLENS, M. 2006. Cognitive wireless networks: your network just became a teenager. In *Proc. of IEEE INFOCOM*.
- MAI, C.-H., HUANG, Y.-C., AND WEI, H.-Y. 2010. Cross-layer adaptive H.264/AVC streaming over IEEE 802.11e experimental testbed. In *Proc. of IEEE VTC Spring*.
- MOVE NETWORKS. 2008. Internet television: Challenges and opportunities. Tech. rep., Move Networks, Inc. November.
- NETVIEW TECHNOLOGY. 2010. <http://www.netview.no/index.php?page=downloader>.
- NI, P., EICHHORN, A., GRIWODZ, C., AND HALVORSEN, P. 2009. Fine-grained scalable streaming from coarse-grained videos. In *Proc. ACM NOSSDAV*. 103–108.
- PANTOS, R., BATSON, J., BIDERMAN, D., MAY, B., AND TSENG, A. 2010. HTTP live streaming. <http://tools.ietf.org/html/draft-pantos-http-live-streaming-04>.
- REJAIE, R. AND ORTEGA, A. 2003. PALS: peer-to-peer adaptive layered streaming. In *International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*. 153–161.
- RIISER, H., HALVORSEN, P., GRIWODZ, C., AND HESTNES, B. 2008. Performance measurements and evaluation of video streaming in HSDPA networks with 16QAM modulation. In *Proc. of IEEE ICME*. 489–492.
- RIISER, H., HALVORSEN, P., GRIWODZ, C., AND JOHANSEN, D. 2010. Low overhead container format for adaptive streaming. In *Proc. of MMSys*. 193–198.
- SCHIERL, T., DE LA FUENTE, Y. S., GLOBISCH, R., HELLGE, C., AND WIEGAND, T. 2010. Priority-based media delivery using SVC with RTP and HTTP streaming. *Multimedia Tools and Applications (MTAP)*, 1–20.
- SCHWARZ, H., MARPE, D., AND WIEGAND, T. 2007. Overview of the scalable video coding extension of the H.264/AVC standard. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 9, 1103–1129.
- SUN, J.-Z., SAUVOLA, J., AND RIEKKI, J. 2005. Application of connectivity information for context interpretation and derivation. In *Proc. of ConTEL*. 303–310.
- TAMAI, M., SUN, T., YASUMOTO, K., SHIBATA, N., AND ITO, M. 2004. Energy-aware QoS adaptation for streaming video based on MPEG-7. In *Proc. of IEEE ICME*. 189–192.
- WAC, K., VAN HALTEREN, A., AND KONSTANTAS, D. 2006a. QoS-predictions service: Infrastructural support for proactive qos- and context-aware mobile services (position paper). Springer Lecture Notes in Computer Science Series, vol. 4278. 1924–1933.
- WAC, K., VAN HALTEREN, A., AND KONSTANTAS, D. 2006b. QoS-predictions service: Infrastructural support for proactive QoS- and context-aware mobile services (position paper). In *Proc. of OTM Workshops*. 1924–1933.
- ZAMBELLI, A. 2009. Smooth streaming technical overview. <http://learn.iis.net/page.aspx/626/smooth-streaming-technical-overview/>.
- ZINK, M., KÜNZEL, O., SCHMITT, J., AND STEINMETZ, R. 2003. Subjective impression of variations in layer encoded videos. In *Proc. of IWQoS*. 137–154.

# Online Appendix to: Video Streaming Using a Location-based Bandwidth-Lookup Service for Bitrate Planning

HAAKON RIISER, TORE ENDESTAD, PAUL VIGMOSTAD,  
Netview Technology AS, Norway

CARSTEN GRIWODZ and PÅL HALVORSEN,  
Department of Informatics, University of Oslo, Norway & Simula Research Laboratory, Norway

## A. CONFIGURATION OF THE BUFFER-BASED REACTIVE ALGORITHM

The reactive algorithm described in section 4.2 has several parameters that can be tweaked to improve its performance. In this section, we present the results from experiments showing how the algorithm behaves when changing the parameter values.

### A.1 Bandwidth capping

If only the amount of data in the buffer determines which quality level to use, the chosen quality level can have a higher bitrate than the current download bandwidth. This is usually not beneficial for a reactive algorithm, because it cannot predict if a future increase in bandwidth will make it safe to drain the buffer. Thus, limiting the chosen video quality to the observed bandwidth help avoid many buffer underruns. Here, the current download rate is calculated using a moving average, sampled every second, where the last sample counts for only a fraction  $\varepsilon$  of the estimate. This prevents strong fluctuations in the bandwidth estimates. Figure 14 shows the effect of capping quality level selection to the current download rate using an  $\varepsilon$  of 10 % and 50 %. We see that, by using the capping, several buffer underruns are avoided, and the number of quality changes is reduced. We also observe that by weighting the last segment download rate less, we further reduce the influence of bandwidth fluctuation, and we therefore use an  $\varepsilon$  value of 10 %.

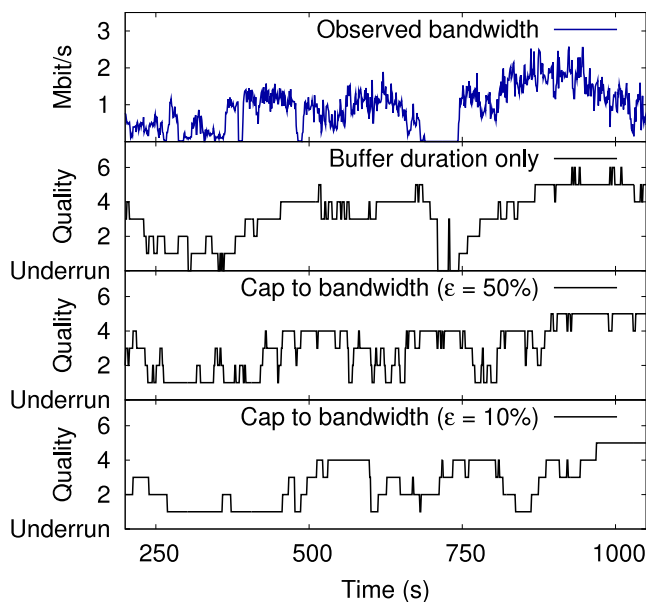


Fig. 14: Cap quality selection effects

## A.2 Scaling the buffer step size relative to video bitrate

The reactive algorithm upgrades the quality once the buffer duration reaches certain chosen thresholds. However, a problem is that, to make a difference in quality in the higher quality levels, the bitrate must often increase dramatically. As shown in table I, the difference between quality levels 5 and 6 is 1500 kbit/s, while the difference between levels 1 and 2 is 250 kbit/s. The thresholds for jumping between layers should therefore take the bandwidth difference between the layers into account.

We propose the following buffer threshold algorithm: When the buffer is empty, the lowest quality (level 1) is selected. When  $B$  seconds of video are buffered, level 2 is chosen. The buffer requirement for level  $N$  is  $B \cdot (R_N - R_1)/(R_2 - R_1)$ , where  $R_N$  is the bitrate of quality level  $N$ .

Figure 15 compares four different settings. The first two use a fixed step size  $B$  of 2 and 10 seconds between all quality levels, i.e., no bitrate scaling. The figure shows that a low step size leads to rapid buffer drainage caused by frequent jumps into quality level 6, leading to several buffer underruns. Increasing the step size to 10 seconds helps, but quality level 6 is still chosen too frequently. The last two settings in the figure use the bitrate-scaling formula described above, with two different base step values ( $B = 2$  and  $B = 10$ ).

Figure 15 shows a large improvement when enabling bitrate scaling and the larger base step size. Using these settings, we get a much better viewing experience, almost removing the buffer underruns and having a more stable video quality.

Note that combining scaled step sizes with bandwidth capping reduces the impact of the scaled step size. With bandwidth capping, the higher quality levels are often disregarded completely, and it is with the higher levels that the scaled step size is most important. However, making it slightly harder to reach the higher layers further improves the results from the bandwidth capping, and we therefore additionally use the buffer step size scaling with  $B = 10$ , even when bandwidth capping is enabled.

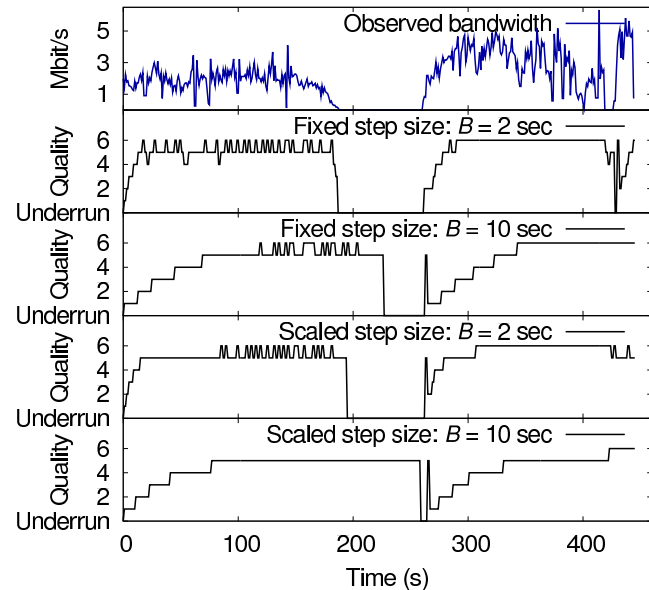


Fig. 15: Altering the buffer step size

### A.3 Removing rapid oscillations in quality

Another goal when improving the viewing experience is to avoid rapid oscillations in quality, as they cause a flickering effect that greatly reduces perceived quality [Zink et al. 2003; Ni et al. 2009].

An easy way to limit this is by having slightly different step sizes when going *up* in quality than when going *down*. Figure 16 shows how requiring 20% more in the buffer when going up than down can avoid having the quality level oscillate between two quality levels (we have also tried percentages both smaller and larger than 20, but the differences are small, and the current value of 20% seems to be a good tradeoff). Since we only require 20% extra in the buffer, it has a negligible effect on how fast the quality increases. Still, it is sufficient to achieve our desired effect. The important thing is simply that we avoid tipping back and forth between two levels when the buffer fullness is exactly on a step size “edge” between two levels. Consequently, we use different thresholds for going up compared to going down in quality.

Additionally, we have found that a slight improvement can be made upon this by also delaying quality upgrades for a few seconds after a drop in quality. We have chosen to delay with 20 seconds, as this proves to be a good trade off between too frequent changes in quality and the aggressiveness of quality selection.

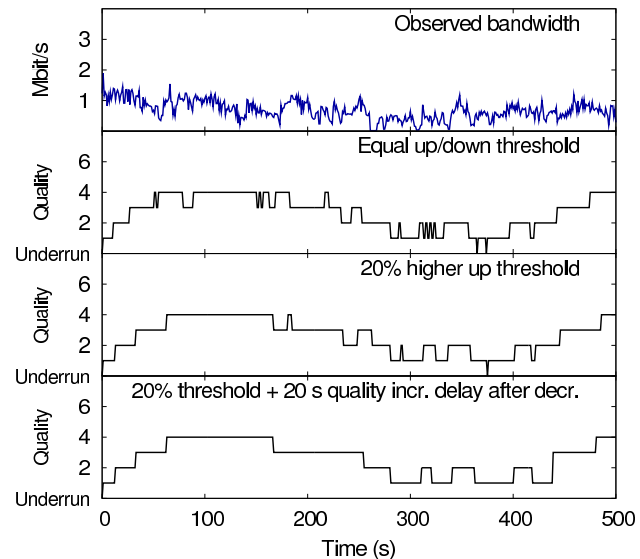


Fig. 16: Removing rapid oscillations

## B. CONFIGURATION OF THE HISTORY-BASED PREDICTION ALGORITHM

The history-based prediction algorithm described in section 4.2 uses the bandwidth lookup service to predict available bandwidth using the average of previous bandwidth numbers. If all streaming sessions would perform as the statistical data from the lookup service indicate, we should be able to use the lookup data directly. However, several factors that may influence the results in a real environment, and there are often mis-predictions leading to buffer underruns. Here, we present the results from experiments showing how the algorithm behaves when tweaking different parameter values in order to find a usable configuration suitable for a range of different situations.

### B.1 Combination with the reactive algorithm

We have found that poor (or plainly wrong) predictions, which are indicated by buffer underruns, make it extremely hard to find adjustments that improve it. Such adjustments break the average case, leading to pessimistic solutions with poor average quality or even more buffer underruns. On buffer underruns, we have tried using a penalty parameter that lowers the prediction based on the variance of measured bandwidths. By making the prediction less optimistic, we could usually recover quickly from a buffer underrung. The problem is resetting the penalty to its neutral setting (i.e., average prediction). Almost without exception, this resulted in more buffer underruns soon after recovery, suggesting that

runs that encounter problems are likely get problems in the future, even after things seem to be back to normal. Thus, we have abandoned this approach and instead use the neutral settings in combination with the reactive algorithm at all times. We let both algorithms select a quality level, and choose the lower one. This means that the reactive algorithm functions as a safety net (only) when the buffer is small.

The reactive algorithm is configured as described in section A, except that we have disabled the bandwidth capping. Bandwidth capping is harmful when having prediction, as it would cause drops in quality even with a large buffer. Because we have a prediction of the future bandwidths, we do not have to worry about short drops in bandwidth. We only need the reactive algorithm when the buffer becomes too small for blindly trusting predictions. Figure 17 shows the benefit of this approach in a streaming session with large negative differences between the observed and predicted bandwidths.

We can clearly see that a huge number of buffer underruns are prevented by enabling the reactive algorithm.

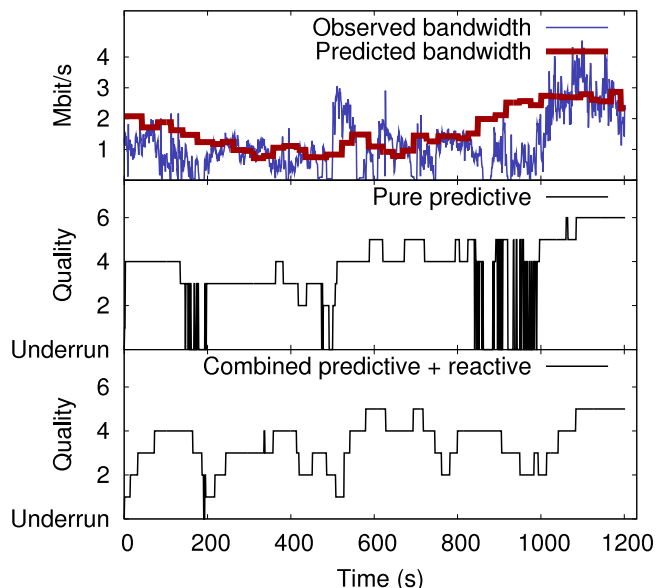


Fig. 17: The reactive algorithm safety net

### B.2 Variable-length travel handling

An omniscient algorithm would know *exactly* how long a trip takes and *exactly* the amount of available bandwidth at all times. This enables it to end the streaming session with empty buffers and maximum average quality. In reality, this is obviously not possible, and a challenge is to handle variable-length travels. Consequently, a real implementation that tries to end with empty buffers takes a great risk, as the margins are very small near the end, and any deviation (delay) from the predicted trip can cause a buffer underrun. Our metro test case demonstrates this, as shown in figure 18. What happens near the predicted end of the trip is that the algorithm ramps up quality dramatically to drain its buffers, in an effort to maximize average quality. When a delay occurs inside the tunnel (where the available bandwidth is almost non-existent) the buffers run

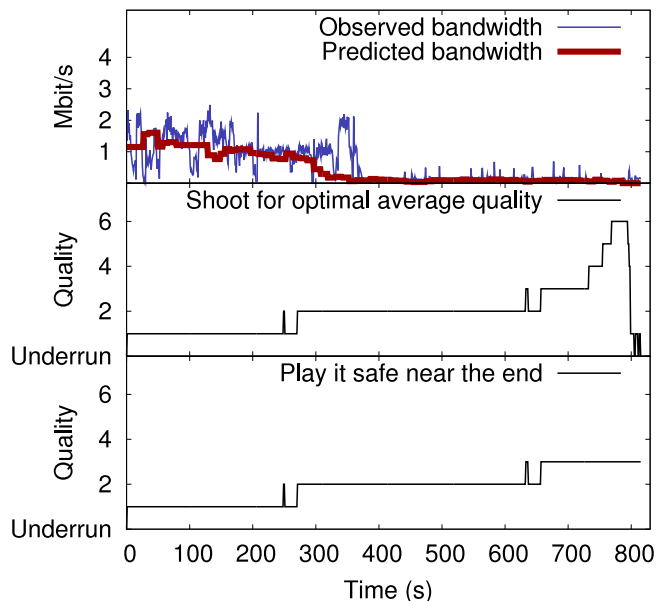


Fig. 18: Limit aggressiveness at the end of the trip

dry before the trip's end, and we are unable to recover due to the lack of bandwidth. This demonstrates an unfortunate special case that needs to be handled.

As a countermeasure, we relax the buffer drain requirements near the end of the trip. When most of the trip is done, there is little point in maxing out the quality just for the sake of optimization, so we limit the quality to the maximum achieved at 85 % of the trip's estimated duration.

The result is that our buffers contain some unused data at the trip's end, but we are better protected from forced bandwidth drain at bandwidth-poor locations near the end.