Scalable Video Coding in Heterogeneous Conference Scenarios

Fabian Jäger

Projektarbeit 1

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Topic of this paper

Scalable Videocodec in a Video Conference

Keywords

Videoconferencing, SVC, scalable video codec, bandwidth estimation, adaptive codec, iPhone, mobile video conference, Daviko, Placecam, H.264

Abstract

Video conferencing over IP (VCoIP) has rapidly spread in the mobile realm, where it faces the problem of heterogeneous, fluctuating network conditions. Scalable video coding enables bandwidth adaptation, but requires guidance by appropriate resource estimators.

This work focuses on the analysis and design of an adaptive, bandwidth-aware transmission strategy for real-time multimedia applications like video conferencing. We present an early indicator of network congestion based on jitter variation along with our implementation of a new lightweight sender-based approach to adapt the video codec.

Contents

List of Figures

1 Introduction

Modern multimedia applications like video conferences are capable of producing high quality video streams. However, the data demands of the video stream often reach the limit of the available bandwidth and therefore inflicts the QoE (Quality of Experience). The QoE is very important in a video conference and can be optimized when the quality of the video stream is adapted to the available bandwidth.

It is preferable to scale the datarate and coding complexity of the video for each participant individually [2]. This holds in particular for mobile regimes [3],[1] and can be done with the scalable video codec (SVC [4]), which is an extension to H.264 [5]. Unfortunately the processing power and available bandwidth of each participant may differ in a video conference.

SVC makes it possible to reduce the amount of data for clients with low bandwidths by dropping enhancement packets, which are not necessarily needed to decode the video stream. The aim is to provide every client with as much data as possible without congesting the link. However, it is neither easy to identify the available bandwidth nor to determine a congested link. In this paper we will discuss a new approach, that uses the jitter variation as a congestion indicator and can be used to adapt the codec to the network conditions. Further, we will test this approach in a multimedia streaming application and present the results.

The paper is organized as follows. The second chapter discusses the problems we have in a video conference. The third chapter introduces our approach of determining a congested link. In the fourth chapter we are going to discuss a basic implementation of the approach, which is already implemented. Our approach is described in the fifth chapter. Our testbed and sample application is presented in chapter six and seven. The eight chapter contains the measurement results without an adaptive code, while the ninth chapter contains the measurement results with an adaptive codec. The tenth and last chapter contains a conclusion and an outlook.

2 Problem Section

The QoE in a video conference is often limited by the network. With SVC we are able to adapt the video codec to the available bandwidth, which can optimize the QoE. However, it is not easy to find a continuous and accurate measure of the available bandwidth, even though a lot of research concerning this topic exists. It is complicated to determine the effective bandwidth of a path, especially when the available bandwidth is not constant. Common approaches use PGM (Probe Gap Model) or PRM (Probe Rate Model) to measure the available bandwidth, which requires extra probing packets [6]. The accuracy of bandwidth estimates depends on the amount of probing packets, the nature of side traffic and the duration of the measurement. Overall, these techniques are intrusive and rather slow and therefore do not fit the real-time requirements of a video conference. The speed of the measurement is an important factor for a bandwidth-aware video scaling. If the measurement is too slow, we are not able to adapt the codec fast enough and the link congests. For a good QoE we reduce or increase the quality of the video stream, depending on link conditions. This decision has to be made as fast as possible and the quality reduction needs to be significant enough in order to avoid a congestion. On the other hand, the video reduction should not be too eager to avoid an overreaction, which also decreases the QoE. Altogether, in a video conference a fast bandwidth estimation is favored over a more accurate but slow approach.

3 An Early Indicator of Network Congestion

Measuring the available bandwidth on a path is not easy and rather slow. Therefore, instead of doing so, we try to predict a congestion on a link and take immediate action to avoid it [1].

A very simple indicator for a congestion is the the size of the sending queue, but this information is not always available (RTCP for example). Also it is too slow when the bottleneck is far away from the sender. In this case, it would take some time until all queues on the routers between the sender and the bottleneck are filled, before the packet queue of the sender gets filled. Still we can use this indicator in some situations where our jitter observation fails. This will be explained in Section 3.2.

3.1 Network Congestion Indicators

For a fast video adaptation we obtain indirect information about the available bandwidth by observing the link and analyzing the delay, jitter and jitter variation on the transport layer. This can be done at the sender or the receiver side. On the receiver side, the one-way delay could be used, but it requires synchronized clocks and is an additional effort. Also the receiver must inform the sender about the link conditions, which requires additional packets and increases the overhead. A sender sided approach uses the RTT and avoids those issues, but may be a problem, if the link is asymmetric congested. In this case we do not know the one-way delays. We use a light-weight sender sided approach, because we assume that links congest symmetrically most of the time and we have a better performance due to the reduced overhead. Even if the link is asymmetrically congested, we only loose accuracy for congestion on the way from the receiver to the sender.

The approach is based on the idea that a congestion abnormally changes the fluctuation of the RTT, which mainly consist of the following components:

Capacity: The capacity is the theoretical maximum data transmission on a path. The capacity inflicts the RTT depending on the packet size. The capacity on wired links varies only if the routing changes and the packets take another path to the receiver. Since this is not very likely to occur, the capacity on wired links practically stays the same at all times. On wireless links the capacity is not constant and may vary.

Path Traversal Time The path traversal time is the time a packet needs to be transmitted on a path. The path traversal time depends on the topological distance between the sender and the receiver. As long as the path does not change, the delay of the pure transversal of the empty path will stay the same.

Queueing Delay: The queueing delay is the delay caused by the routers on a path. Every router on a path adds a delay to the RTT, which mainly depends on the amount of packets in the queues.

Every congestion adds significant queueing delay to the RTT which differ from the regular RTT fluctuations. Even though we are trying to detect these changes, we cannot use the RTT directly for a fast congestion detection, since the RTT mainly depends on the topology and a high RTT does not necessarily indicate a link congestion. On a congested link, router buffers start to fill and the RTT rises. Still it is complicated to differentiate between a congested link and regular RTT fluctuations as characterized by the jitter. The jitter is used to differentiate between a regular RTT fluctuation and a congestion. The congestion is identified by a significant RTT increase, which exceeds the regular jitter tolerances. This increasing jitter goes along with a jump in its derivative, the jitter variation. We observe the jitter variation and use this discontinuous jump as an indicator for a congestion if the RTT is rising. If the RTT shows a negative drift, the congestion is resolving.

3.2 Challenges in Parametrizing Indicators

The decision whether and how to react to an increasing jitter variation is important for a good QoE.

W_{in}	Bandwidth Competing Traffic
U_{in}	Bandwidth Video Stream
W_{out}	Available Bandwidth

Table 1: *Demand values for the link*

Table 1 shows the demand values which are needed to steer the congestion identification. The link is congested if $W_{in} + U_{in} > W_{out}$. Since we have no influence on the competing traffic W_{in} and the available bandwidth W_{out}, we scale the video stream until $W_{in} + U_{in} <= W_{out}$.

Video Quality	Bandwidth Competing Traffic
Measurement Time	Time we need to gather network information before we can react
Jitter Variation Threshold	A threshold for the jitter variation to avoid reactions to normal fluctuations

Table 2: *Parameter for the congestion control*

Table 2 shows the parameter we have to steer the congestion control. The quality of the video is adapted to the jitter variation. If a congestion occure, the quality is reduced to lower the bitrate of the video. The precision of the jitter variation measurement depends on the measurement time. A short period of measurement allows a fast reaction, while a long period of measurement provides more precise values. To avoid unnecessary reaction to normal jitter variation fluctuations, we use a threshold.

The jitter variation threshold may be a problem in some situations. A very modest congestion of a link, where $W_{in} + U_{in}$ is just a little higher than W_{out} results in a slowly increasing RTT. In this case, the link would congest slowly, but it gets harder to differentiate the jitter variation from a normal fluctuation, and if it were below the threshold we would not react.

Another problematic situation arises, whenever we react insufficiently to an increasing jitter. In some cases, a sudden and severe congestion occurs and it would be the best to instantly reduce U_{in} as much as possible by decreasing the quality. If we take too much time to reduce U_{in} or if the reduce of quality is insufficient, we will still congest the link. In this situation, it is complicated to resolve the congestion, since we recognize a congestion by an abnormal fluctuation of the RTT. The longer the congestion takes, the harder it gets to differentiate between a congested link and regular RTT fluctuation, because we are losing the reference to the RTT fluctuation on the free link.

A solution to these problems could be the observation of the sending queue size if this information is accessible, because every congestion results in an increase of the sending queue. This might take more time (especially when the bottleneck is far away from the sender) than the observation of the jitter, but it is more precise since. This situation only occurs if $W_{in} + U_{in} > W_{out}$, which indicates that the receiver is overloaded or the link is congested. Whenever the jitter observation fails to work as intended, we hope to notice this on the sending queue size and be able to react to the congestion anyway.

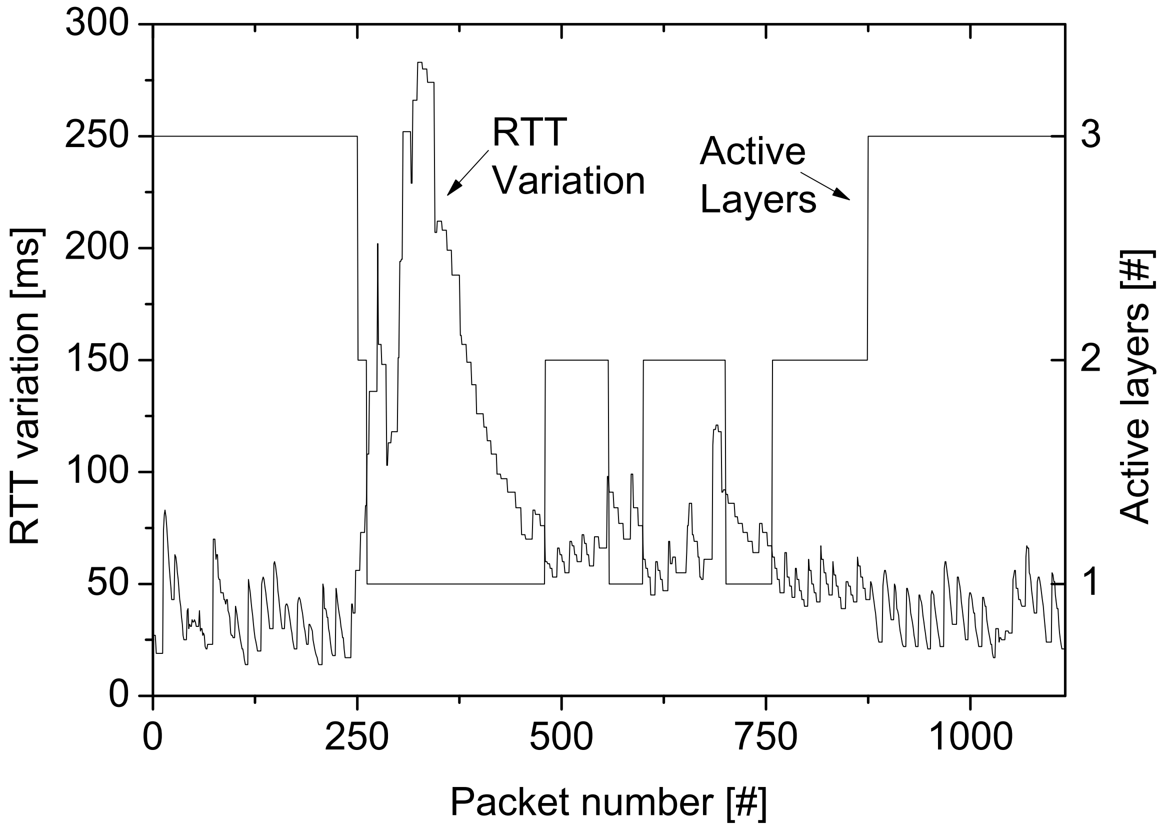

Figure 1: *Videolayer Adaptation taken from [1]*

4 Basic Implementation of the Approach

Previous works with a very basic implementation of the new approach adapt the video quality to the available bandwidth using a very simple mechanism. It is already implemented and used in a video conferencing software. In this section we will describe the approach, before we will present our approach in Section 5.

The basic idea is to observe the jitter on a link and to lower the video quality in case it increases over a period of 10 subsequent packets or raise the quality in case it decreases over a period of 50 subsequent packets. The reason for the different amount of packets is to ensure a fast reaction, if the link gets congested and a cautious quality increasing. Currently, three temporal layers are used to adapt the codec to the available bandwidth [1].

4.1 Videoscaling: Current State

A simple scenario including all three layers is shown in Figure 1. This means that the quality of the video stream is at a maximum and the most bandwidth is being used. After 250 video packets, the variation of the RTT (jitter) begins to rise. After 10 packets with an increasing jitter, the application removes the additional layer and only the base layer remains. After 500 packets the jitter decreases over a period of 50 packets and the application adds an enhancement layer. Nevertheless, the link will still appear congested because the amount of layers is unstable until the transmission of 750 packets, at which point an enhancement layer is added and after 875 packets all three layers are added.

4.2 Potential Optimizations

Starting from this simple approach, the following options for improvement can be considered. First, the period to measurement is kept constant (10 and 50 packets), but depends on the packet rate. This may be a problem, if the link is very unstable, because in this case a duration of 10 packets might be too long to react fast enough, or 50 packets might be too soon to increase the video quality. In both cases, the link congests. On the other hand, there are situations in which 10 packets is too short to react with a quality reduction or 50 packets too slow to react with an increased quality. In these cases, we would not use the available bandwidth optimally. It would be better, if the measurement points were variable and could be adapted to the characteristics of the link or the video stream.

Second, the measuring time based on the transmitted packets could be improved, since the interarrival gap between two packets is not constant, but relies on the conditions of the link. If the link is congested, the packets need more time to be transmitted, which increases the interframe space. Therefore the transmission time for 10 packets on a congested and 10 packets on a congestion-free link differs. This issue complicates the implementation of a solid and comprehensible video adaptation algorithm. Therefore, we suggest to use fixed measuring points in time based on the frame-rate of the video stream.

Finally, we would like to point out that this approach only uses temporal layers to scale the video codec. The adjustment of the video quality is limited in this current approach, due to the fact that only 3 temporal layers are used. This may be improved by adding quality or spatial layers to increase the granularity of the video scaling.

5 Video Stream with an Adaptive Video Transmission Control

SVC codecs use multiple temporal, spatial and quality layers, which allow to vary the quality of an encoded video stream. In contrast to common codecs, the SVC always encodes the video with the highest quality and the sender scales the video stream for each participant individually by adding or removing enhancement layers. However, external information, for example the available bandwidth, are needed to control scaling. With the jitter-based prediction, we are able to detect a congested link and to adapt the video stream.

In this approach, we do not know the available bandwidth on a link at first. This complicates scaling, since we cannot set the codec to a correct bitrate. Instead, we have to adapt the video stream to runtime conditions as measured by the jitter behaviour discussed in Section 3.2.

Our approach is able to react to a congested link by reducing or increasing the quality or the temporal layers. Whenever the jitter variation rises significantly and the RTT is increasing, our approach reacts with a reduction of the quality. This can be achieved by a variation of the quantization factor or a reduction of the temporal enhancement layer.

Jitter Variation	The Jitter Variation of the video stream
Avg. Jitter Variation	The approximated Jitter Variation over a period of 8 frames
Jitter Variation Ratio	Ratio of Avg. Jitter Variation and Current Jitter Variation
Quality Layer	The quality of the video stream
Temporal Layer	The amount of enhancement layer (1, 2 or 3)
Jitter Variation threshold	We react only to Jitter Variations above this threshold

Table 3: *Parameter for the Adaptive Video Transmission Control*

The goal of the congestion control is to determine significant jumps in the jitter variation and react to them. In Table 3 an overview of all important parameters is shown.

We observe the jitter variation over a period of 8 frames and calculate the exponential moving average of the jitter variation. The current jitter variation is compared to the average jitter variation and the ratio is used to adapt the codec. Table 4 shows how the congestion control reacts to different ratios.

We also use a 5 ms jitter variation threshold, because the ratio is generally higher, if we compare two low values. For example a 1 ms jump in the jitter from 1 ms to 2 ms would result in a 2.0 ratio, while the same jump from 5 ms to 6 ms results in a 1.2 ratio. At the moment, the 5 ms threshold is an experimentally determined fixed value, which could be improved by a variable value in the future.

Jitter Ratio	Quality decrease
1.0 - 1.2	5%
1.2 - 1.4	5%
1.4 - 1.6	10%
1.6 - 1.8	10%
1.8 - 2.0	15%
2.0 - 2.5	15%
2.5 - ∞	25%

Table 4: *Codec quality adaptation with respect to the Jitter Ratio*

The number of temporal layers in the stream depends on the quality of the video stream. A temporal enhancement layer is added if the quality stays above 60% over a period of 8 video frames. If the quality stays below 40% over a period of 8 video frames a temporal enhancement layer is removed. This strategy is based on the quality and is only indirectly influenced by the jitter variation. Therefore the reaction time with the temporal enhancement layer is much slower, but since changes on the temporal layer inflict the bitrate much stronger, we take more time to decide if it is beneficial.

To increase the quality is more complicated, since we do not have a reliable indicator of a free link. We can assume a resolving congestion if the delay shows a negative drift and then react to it, but whenever the video stream bitrate stays below the available bandwidth and additional bandwidth gets available, the jitter variation does not change. Therefore the quality increases by 5% - 15% (depending on the current quality) when the quality does not change over a variable period of frames, which is initialized with 8 frames. If the jitter variations stays stable, we lower the period of frames by one frame. On the other hand, when the jitter variation changes significantly, the period of frames is reset to 8. This is an optimization to realize a faster quality increasing on a free link. Otherwise it takes too long until the quality is high enough to use the available bandwidth optimally.

6 Network Congestion Indicators in a Streaming Application

We developed a video streaming application to test our approach. A raw video file or a video taken by a webcam is encoded and afterwards sent over the network to one or several receivers. With the video streaming, we gather information about the RTT, the jitter, the jitter variation, the codec and the sending queue size (if accessible). Whenever a congestion

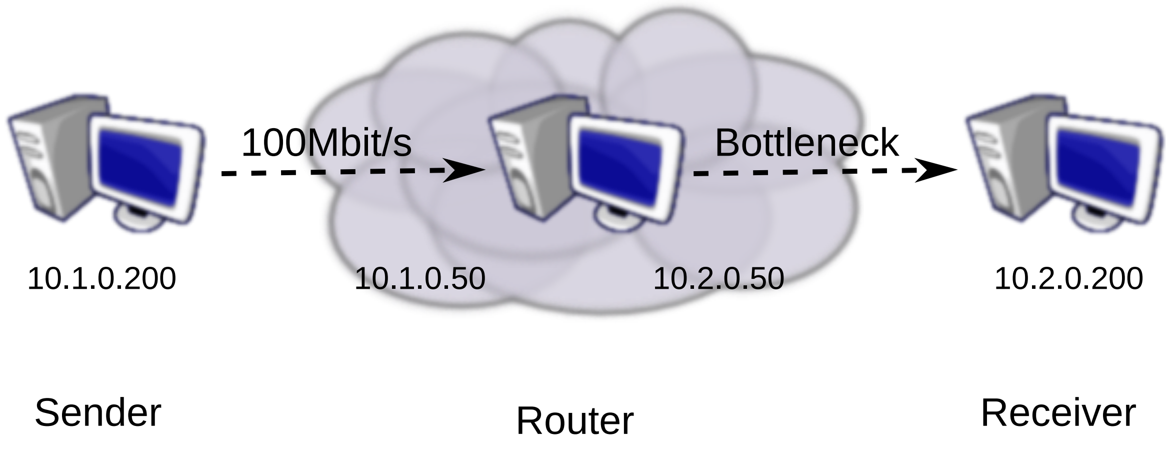

Figure 2: *Testbed*

occurs, we try to detect the abnormal fluctuations in the jitter variation and decrease the quality of the video stream to reduce the bandwidth demands.

We use a reliable transmission of packets in our application, since almost every frame (except I-Frames) of the video stream relies on earlier frames. If one of them gets lost, the whole GOP (Group of Pictures) is corrupted. TCP is not a proper solution, because the Head-of-line-blocking decreases the performance. We therefore use a light-weight protocol on top of UDP called enet [7] for the video transmission. This protocol is developed for real-time applications and provides optional reliable and in-order delivery of packets. The enet protocol also gathers information about the network like the RTT or the jitter and also provides information about internal state of the protocol like the reliable and unreliable packets buffer queue size. Therefore the enet protocol is a good solution for the video stream transmission and it also gives information about the network we need for our approach.

In our implementation, we use the DAVC codec [1] by Daviko [8] that is able to add temporal enhancement layers to the video stream in an H.264-compliant way. We also use the quantization factor of the codec to scale the quality of the video stream. With these two components, we implemented a fine-granular video adaptation. For testing purposes, we use three temporal layers for a rough video scaling, while the quantization is used for a more fine granular adaptation. In this scenario, we do not consider optimization of scaling from an QoE perspective. Instead, we focus on the bitrate of the video stream with the aim of exactly matching the available bandwidth. A scaling that also optimizes QoE will need further research in the future.

7 Testbed

With the streaming application, we are able to test our approach, but we need an environment where we have all information about the links and are able to control the network conditions to verify the measurement results. We used a small network, which is shown in Figure 2, to test our streaming application. It consists of a sender, a receiver and a router connecting these two. We can use the router to scale the available bandwidth on the path and create a bottleneck. The sender and the receiver are desktop computers running Linux and OS X. We use standard hardware and is not further described, since it has only a slight influence on the network, but it should be mentioned that it is necessary for the sender to have enough processing power to encode a video stream fast enough.

Instead of using an external traffic generator, we implemented a simple application, which generates competing UDP/TCP traffic at a given point in time with a variable rate. One benefit is that we have much more control of the generated traffic and are able to change it in real-time. Furthermore, we have information about the sending time as well as the size of a competing packets, which is helpful in order to compare the test results.

7.1 Configuration of the Router

The router uses Ubuntu and the interfaces are configured as followed:

```
#!/bin/sh
ifconfig eth1 10.1.0.50 netmask 255.255.255.0
ifconfig eth2 10.2.0.50 netmask 255.255.255.0

route add —net 10.2.0.0 net mask 255.255.255.0 dev eth2
route add —net 10.1.0.0 net mask 255.255.255.0 dev eth1

echo 1 > /proc/sys/net/ipv4/ip_forward
```

The interfaces have the IP addresses 10.1.0.50 and 10.2.0.50. We also added the routes to the routing table. With the last command we had to enable the IPv4 forwarding on the router, which is disabled by default.

For emulation of a congestion, we used a bandwidth restriction tool called wondershaper [9], which is an easy front-end for tc. The tc tool is the standard tool used for configuring the traffic control in the Linux kernel and it is used for shaping, scheduling, policing and dropping. With these tools we are able to limit the capacity of the up- and downstream for each interfaces.

7.2 Limitations of the Testbed

The purpose of the testbed is to simulate a bottleneck on a link, but the conditions on the testbed differ from larger networks or the Internet.

Since we have only one hop between the sender and the receiver, the bottleneck is always adjacent to the sender. Therefore, we cannot emulate a situation where the bottleneck is far away from the sender. This does not influence the RTT or the jitter, but it influences time until the sending queue gets filled on the sender side. The sending queue gets filled faster when the bottleneck is close to the sender. The RTT is very small on our testbed and therefore the sender can react rather quickly to changes on the jitter. On the Internet, the RTT should be higher and therefore it would take more time until we would be able to react. This may be a problem on links with a fast fluctuating RTT, where a quick codec scaling is necessary to ensure the best possible QoE.

It is difficult to simulate a 'normal' Internet link since they are very multifaceted. It depends on the amount of UDP and TCP connections and especially on the applications with their own characteristic traffic. For example, an HTTP application like a Browser has short TCP peaks when a new site is requested, while a streaming application has a mostly constant UDP

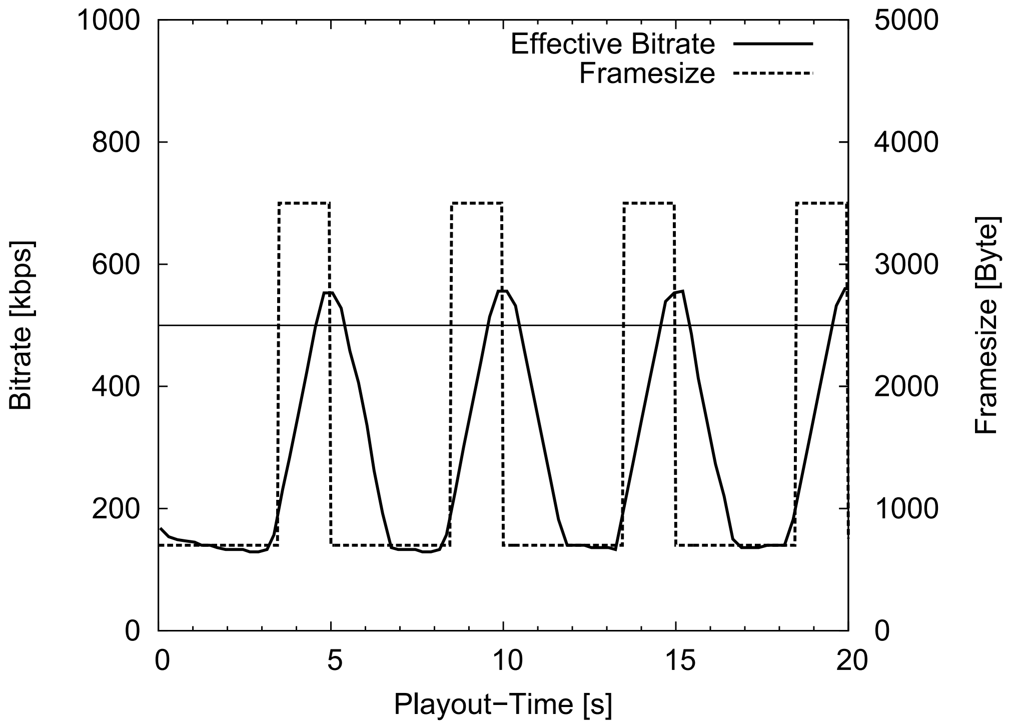

Figure 3: *Square-wave testvideo on a link, which congests if the effective bitrate exceeds the 500 kbps available bandwidth*

stream. The applications might also react to congestion and change their traffic depending on the available bandwidth. This variety is complex to simulate and a lot of related work exist researching this issue. In this work we will focus on simple TCP and UDP test-traffic, because we measure the basic behavior of the jitter variation on congested links and a realistic competing traffic is not necessary.

8 Measurement

To measure the concept of the jitter variation indicator, we generated a simple teststream with huge jumps in the bitrate (square wave) on a network with 500 kbps available bandwidth. The framesize and effective bandwidth of the testsequence are shown in Figure 3 and the results of the measurement are shown in Figure 4. It is show that the RTT and Jitter increase if the

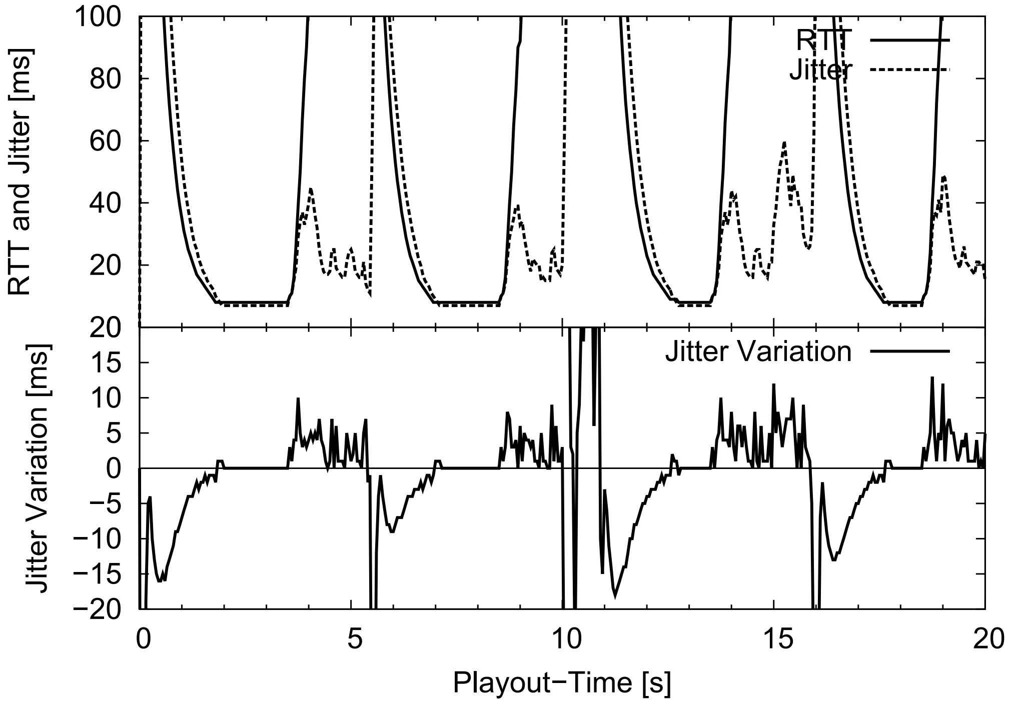

Figure 4: *RTT, Jitter and Jitter Variation of the square-testsequence*

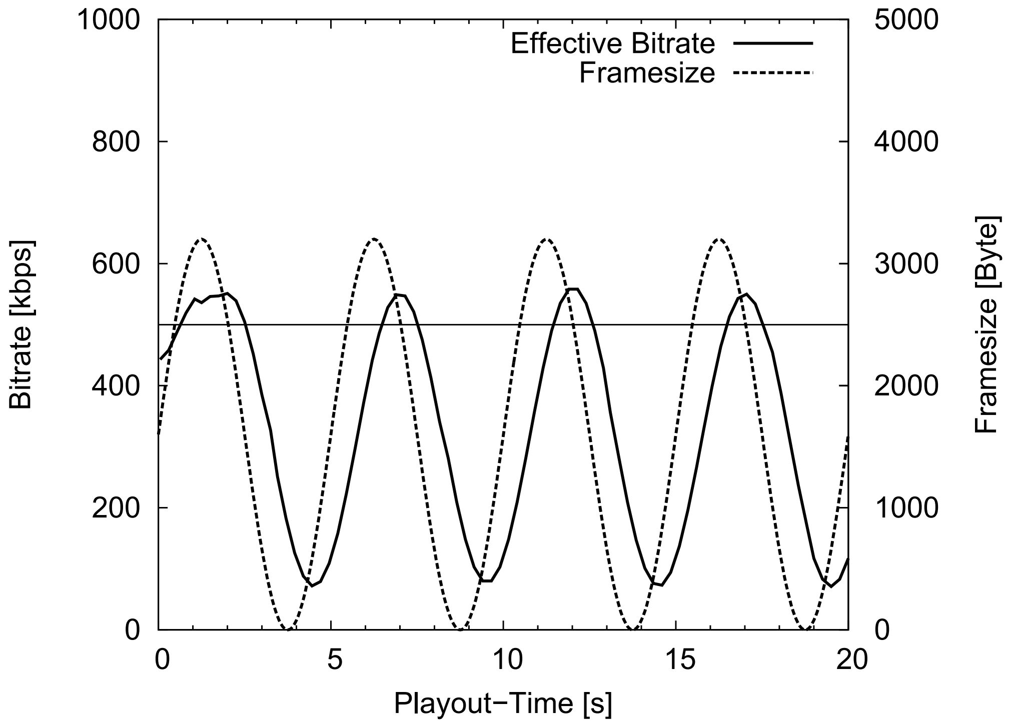

Figure 5: *Sine wave testvideo on a link, which congests if the effective bitrate exceeds the 500 kbps available bandwidth*

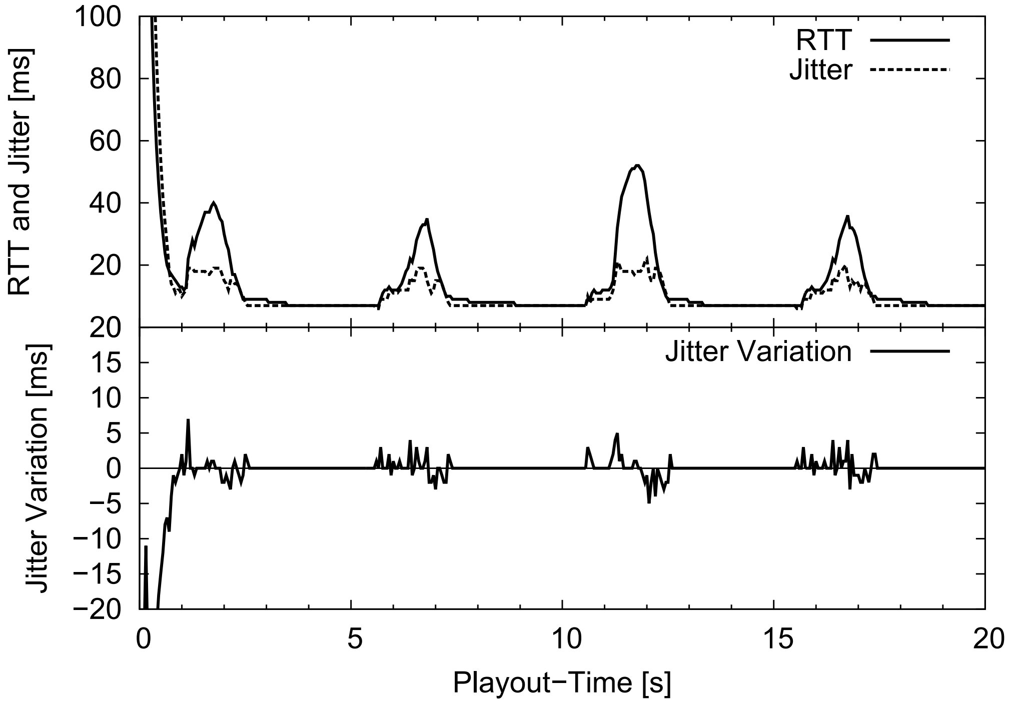

Figure 6: *RTT, Jitter and Jitter Variation of the sinus-testsequence*

framesize is set to 3500 bytes. As soon as the framesize is set back to 800 bytes, the RTT decreases. The changing RTT is also indicated by the Jitter and Jitter Variation. It can be noticed that every time the framesize increases and the approximated bitrate exceeds the 500 kbps available bandwidth, the jitter variation increases significantly. The jitter variation reacts faster to this congestion than the RTT or the Jitter. However, these huge jumps are quite easy to detect and therefore we also used a sine wave as testsequence, which is shown in Figure 5. The results are shown in Figure 6. Even without huge jumps in the video stream, we are still able to observe the exceeding of the available bandwidth in the jitter variation.

For further tests, we used a 15 seconds 768x576 pixel video with 20 fps as test sequence. After 15 seconds of playout-time, the beginning of the video stream is repeated for another 5 seconds, which makes it a 20 second playout-time altogether. The first 5 seconds of the video have to be repeated, because at the beginning the measurement is influenced by the initialisation. The RTT and Jitter are approximated values over a period of 16 packets. Since the enet protocol initializes the RTT and Jitter with 500 ms it takes a short time until we have usable values. We also need this additional 5 seconds in order to analyse the behaviour of the only Intra frame at the beginning of the video stream, which is usually significantly bigger than the Inter frames and therefore an important part in our measurement.

The bandwidth characteristic of the video is shown in Figure 7. The x-axis represents the time in units of s, while the y-axis pictures the bitrate of the video at a certain point in time. As we can see the video bitrate never exceeds 1200 kbps and it is possible to transmit the video on a path with 1200 kbps available bandwidth without congesting the link. Even with 800 kbps, the video has enough bandwidth for most of the time and the link only congests at the 7s and 15 s mark. Below the 800 kbps the link will more often congest, while above the 800 kbps less congestion occur. In this work, we will analyze the RTT, jitter and jitter variation around this bandwidth.

A RTT higher than 200 ms (100 ms one-way delay on a symmetric link) is a noticeable congestion in this set-up and we will use this as an indicator for a congested link [10].

8.1 Video Stream on a Free Link

We measured the RTT, the jitter and the jitter variation on a free path without any congestion (both links had 100 Mbps up- and downstream). The results are shown in Figure 8. The RTT, the jitter and the jitter variation behave as expected on a free path. Sufficient bandwidth is available for our video stream without any competing traffic and therefore a constant RTT, which is not inflected by any queuing delays. However, we have a little peak at the 15s mark. At this point in time, the video starts from the beginning and a new Intra frame is encoded, which is significant bigger than the Inter frames.

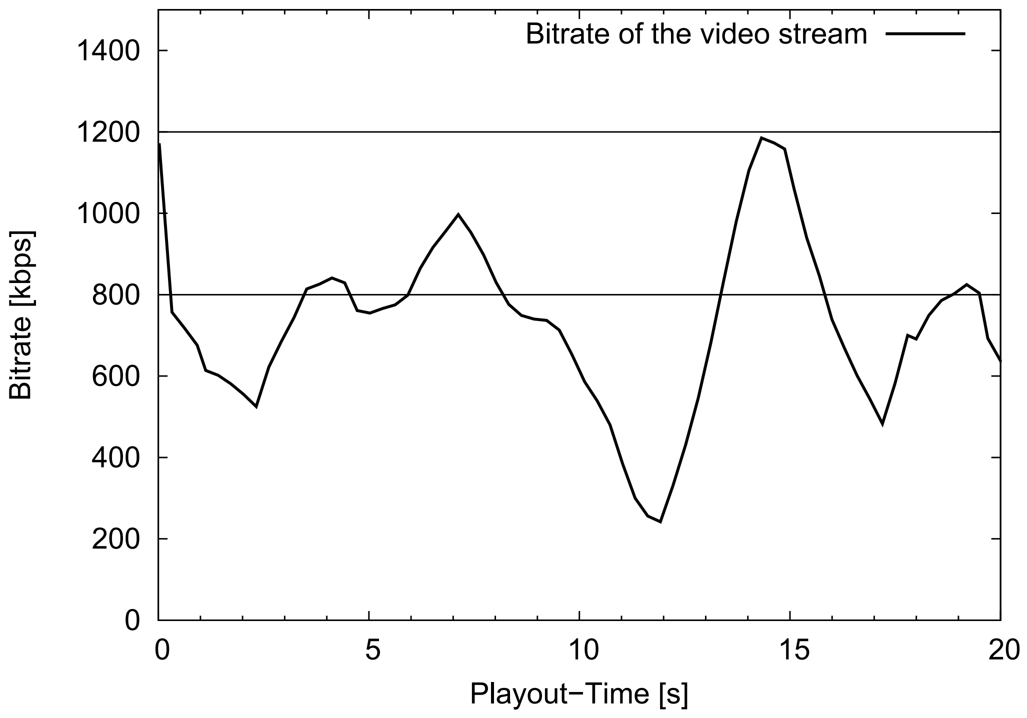

Figure 7: *Bitrate variation for test video*

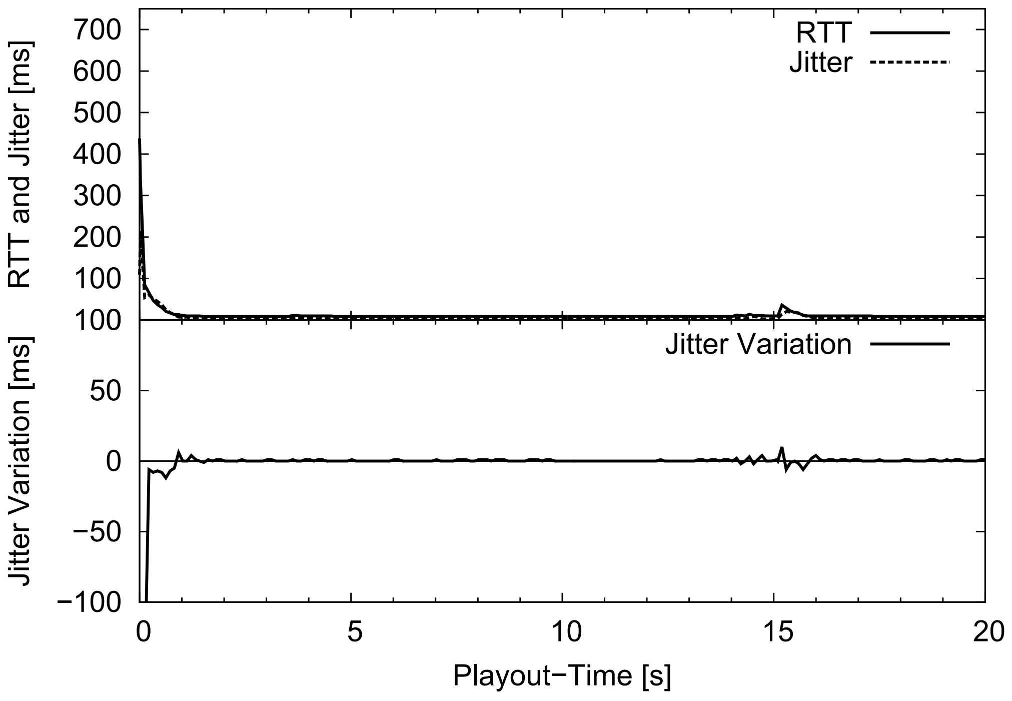

Figure 8: *RTT, Jitter and Jitter Variation on a free link*

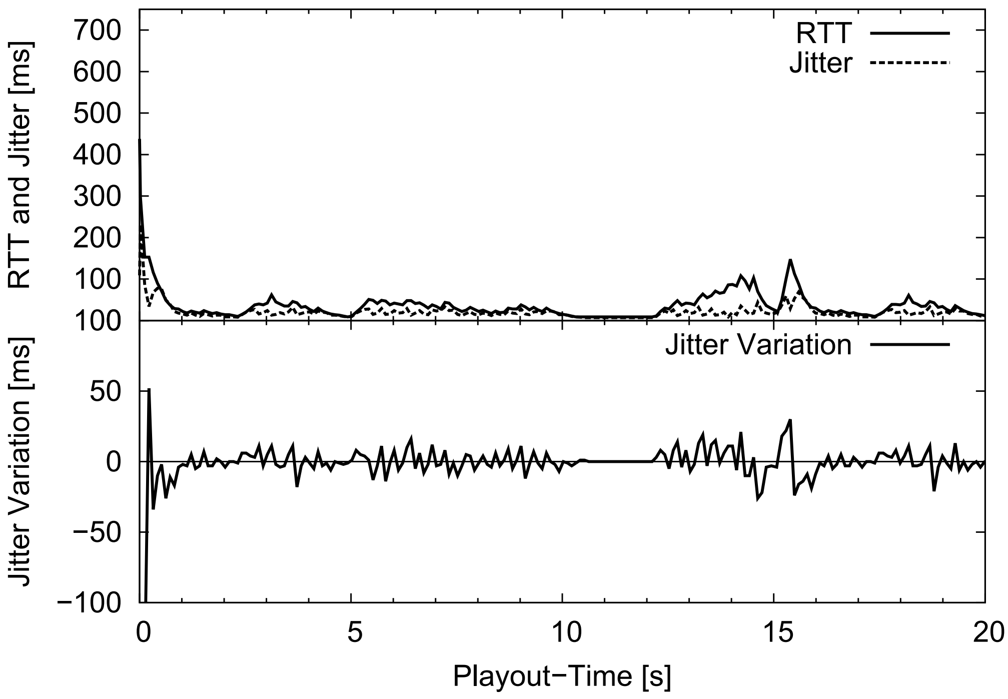

Figure 9: *RTT, Jitter and Jitter Variation on a link with 900 kbps available bandwidth*

8.2 Video Stream on a Constrained Link

In this scenario, the same video with the same settings is measured on a path with restricted bandwidth. The required bandwidth is around 800 kbps and we set the bottleneck to 900, 800 and 700 kbps to analyze the RTT, the jitter and the jitter variation at this rates.

In Figure 9, the available bandwidth is set to 900 kbps and the results differ from the results on a free link in Figure 8. The RTT is not constant anymore, but varies especially at the points in time where the bitrate of the video matches or exceeds the available bandwidth (at 3 s, 5 s and 13 s). Altogether it is still enough bandwidth available and the RTT is low enough to ensure a good QoE.

In Figure 10 we set the available bandwidth to 800 kbps. In general, the results appear mostly similar to the results with 900 kbps available bandwidth, but with a higher RTT at 13 s. The results also show why the jitter variation is a good indicator for a congestion. The rise of

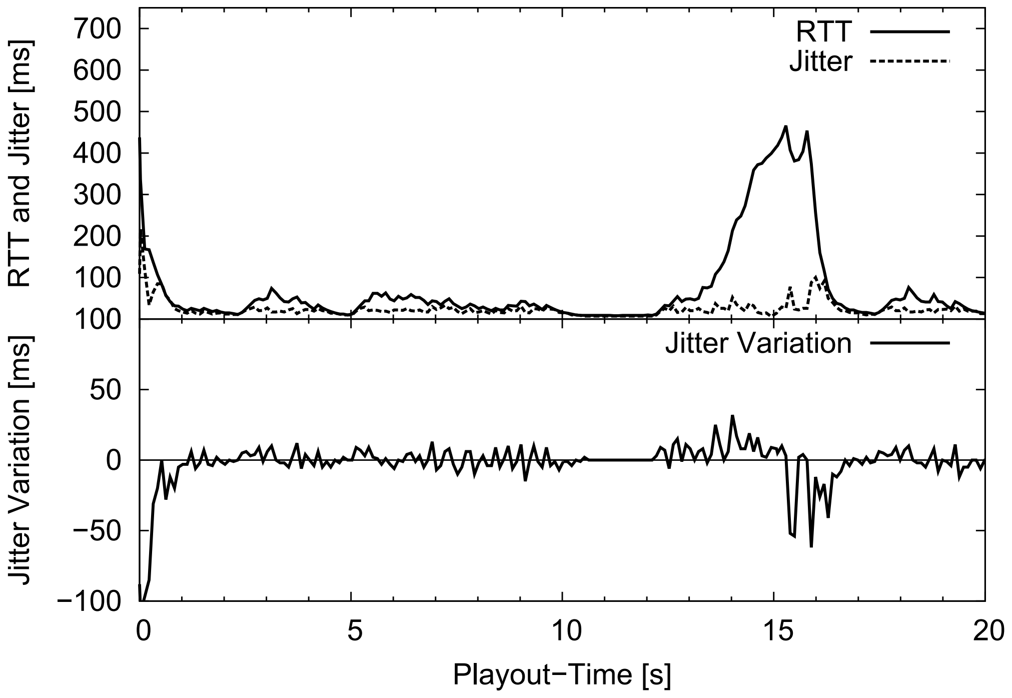

Figure 10: *RTT, Jitter and Jitter Variation on a link with 800 kbps available bandwidth*

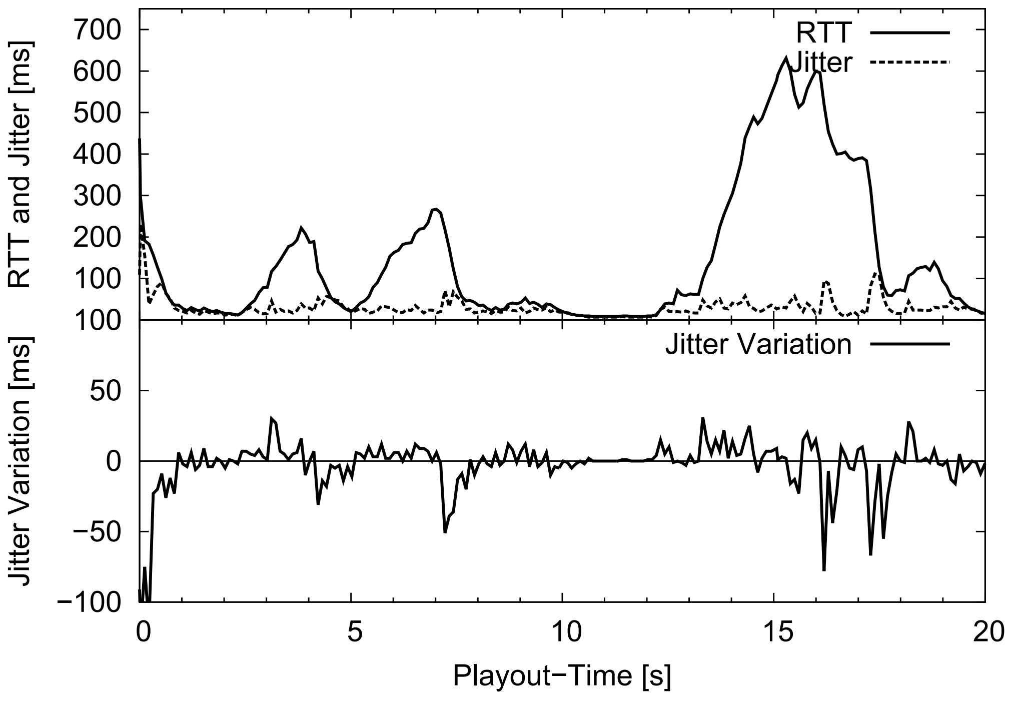

Figure 11: *RTT, Jitter and Jitter Variation on a link with 700 kbps available bandwidth*

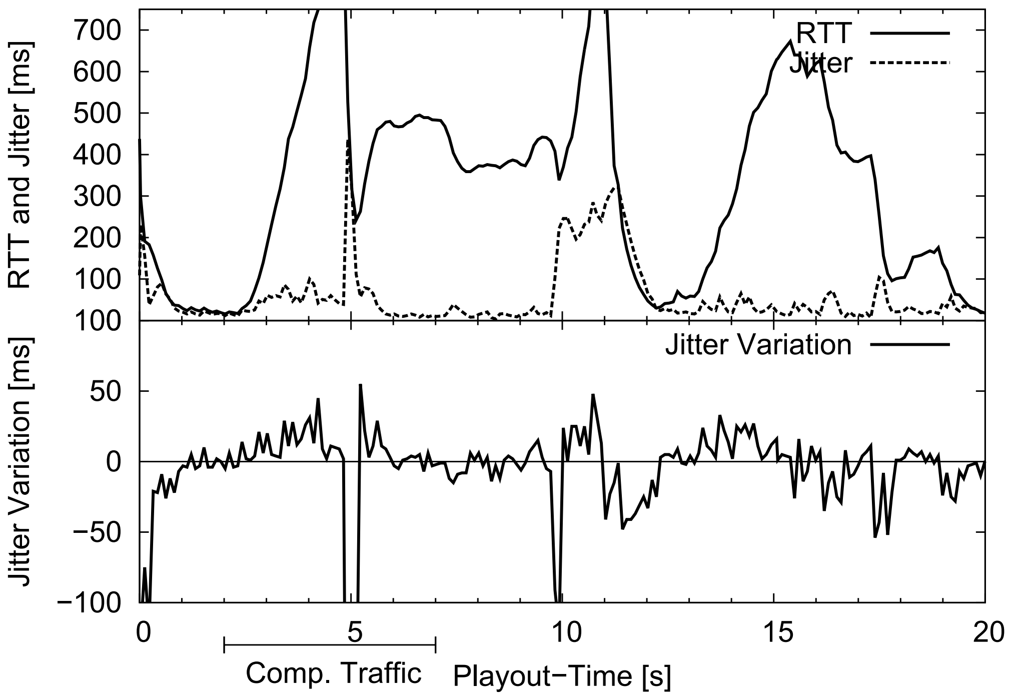

Figure 12: *RTT, Jitter and Jitter Variation on a link with 700 kbps available bandwidth and 100 kbps competing TCP traffic*

the RTT at 13 s goes along with a significant jump in the jitter variation. In general, whenever the link congest and the RTT starts to rise, the jitter variation will jump significantly.

Besides the short congestion we have enough available bandwidth for the video stream in both scenarios (900 and 800 kbps). Figure 11 shows the test results for the video stream on a link with 700 kbps available bandwidth, which is clearly not enough for the video stream. The link congests at the 3,5 and 15 second mark and the RTT is very high. These congestions are noticeable by the user and therefore inflict the QoE. As in Figure 10 at 13 s, we are able to predict the congestions using the jitter variation.

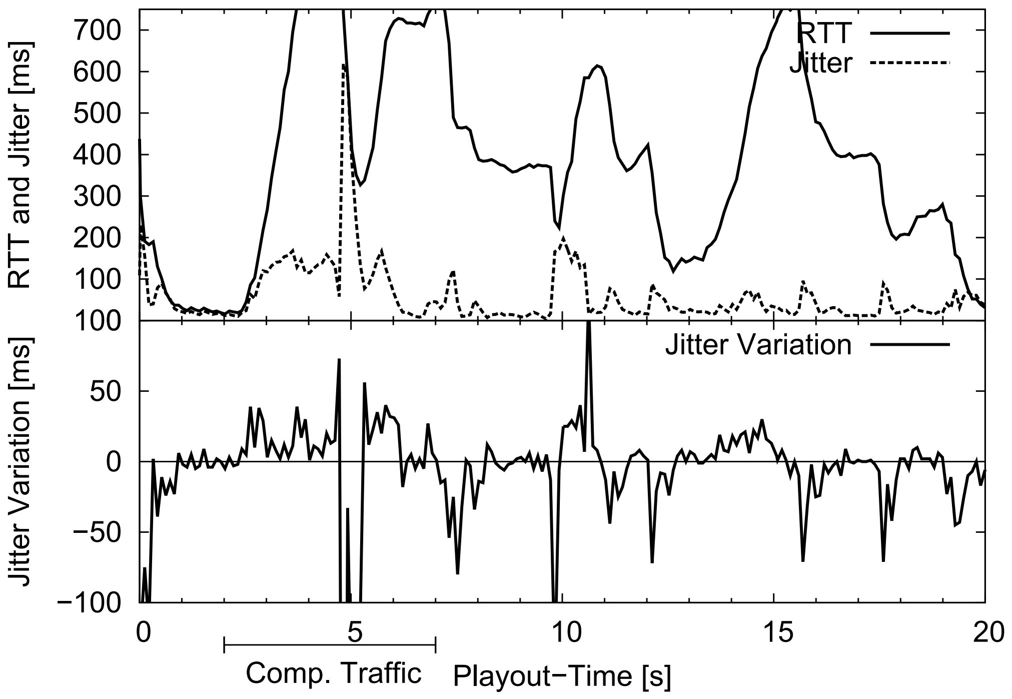

Figure 13: *RTT, Jitter and Jitter Variation on a link with 700 kbps available bandwidth and 100 kbps competing UDP traffic*

8.3 Video stream with Competing TCP Traffic

In this scenario, we have 700 kbps available bandwidth, and generate 100 kbps stream competing traffic at the 2 s mark, which lasts for 5 seconds and send it via TCP. Due to the TCP Flow Control the 100 kbps is not fixed and may vary on the network. Compared to the results from 11 the RTT is a much higher and the congestions last longer. The results are shown in Figure 12. Unlike UDP, the TCP protocol recognizes a congested link and reacts to it with a reduction of its own rate, which is nevertheless not enough to avoid a congestion. The congestion, which is caused by the competing TCP traffic is indicated by the jitter variation and therefore recognizable for our congestion control.

8.4 Video stream with Competing UDP Traffic

Unlike TCP, UDP does not react to link conditions and therefore inflicts the video stream much stronger. We used 700 kbps as the available bandwidth for the congested link (which we already analyzed in Figure 8.2) and added a UDP stream with 100 kbps as competing traffic. The competing traffic starts at 2s and lasts for 5 seconds. The result is shown in Figure 13. In this case our measurement results are heavily effected by the competing UDP stream. In comparison with TCP as competing traffic in Figure 12 the RTT is worse in this scenario due to a non-existing congestion control in UDP.

9 Test Results with Adaptive Video Transmission Control

As in Section 8.4, we use a link with 700 kbps available bandwidth and a 100 kbps UDP competing traffic stream. In this scenario we use the adaptive video transmission control, which is described in Section 5.

9.1 Test Results without Competing Traffic

Figure 14 shows the results for the scaled video stream without competing traffic on the link. Compared to the unscaled video stream in Figure 11, the RTT does not fluctuate much and stays low all the time. No congestion occurs. The adaptation behavior is shown in Figure 15. It is sufficient to scale the video stream with the quality factor and it is not necessary to remove temporal layers to stay below the available bandwidth. A comparison of the scaled video bitrate to the unscaled video bitrate is shown in Figure 16. The scaled video stream stays below the 700 kbps available bandwidth all the time in order to prevent a congestion.

9.2 Test Results with Competing TCP Traffic

The results for the 700 kbps link with additional 100 kbps competing traffic and a scaled video stream in Figure 17 look similar to the results without the competing traffic. However, the competing TCP traffic is visible in the adaptation behavior in Figure 18. Compared to the results without competing traffic in Figure 15, the quality is significantly lower between 2s and 7s, which leads to a lower bitrate of the video stream, showed in Figure 19.

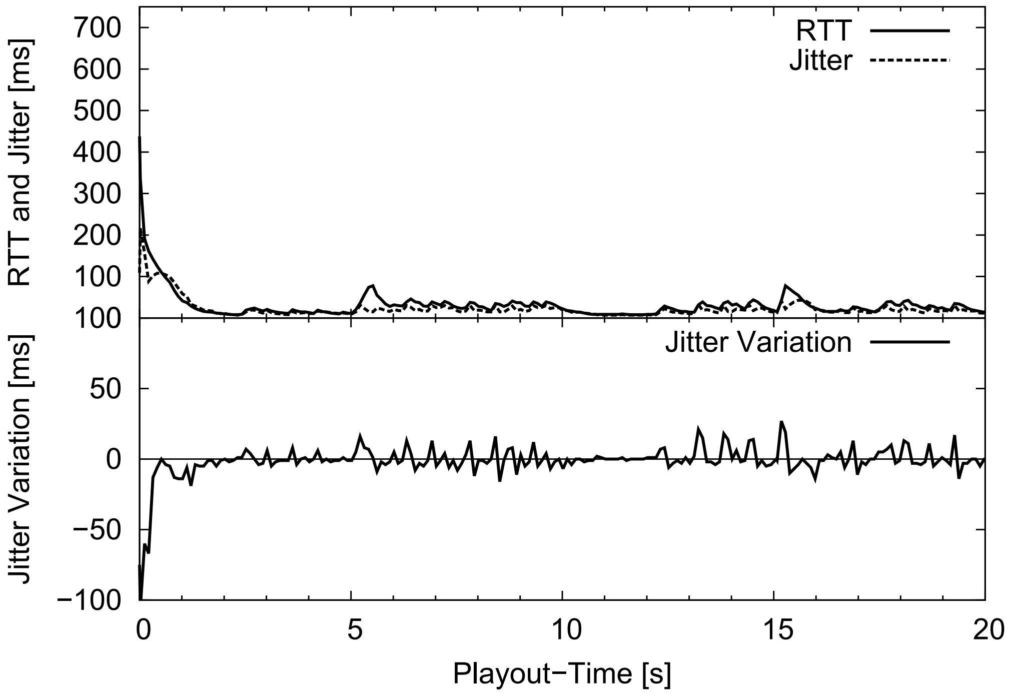

Figure 14: *RTT, Jitter and Jitter Variation on a link with 700 kbps available bandwidth*

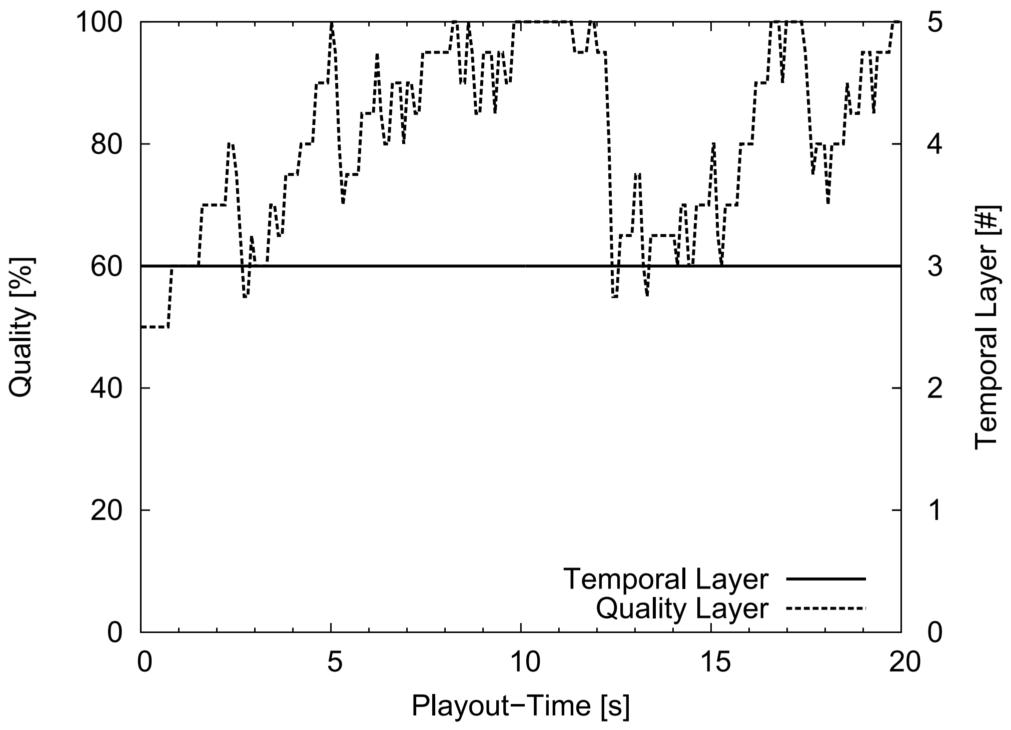

Figure 15: *Quantization and Temporal Layer on a link with 700 kbps available bandwidth*

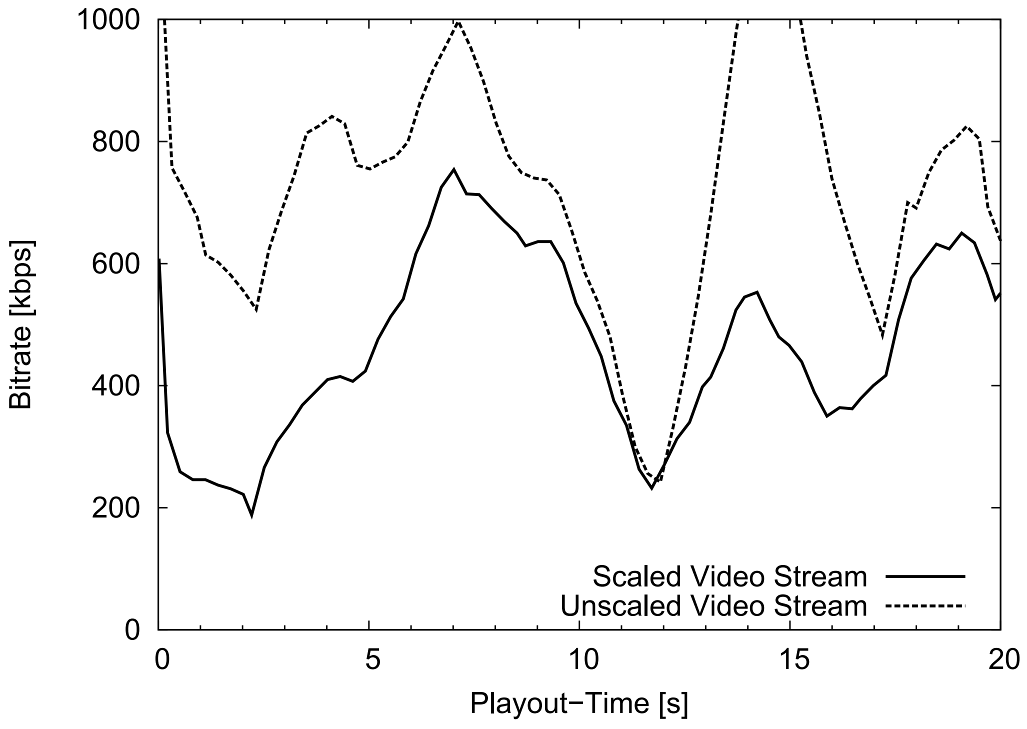

Figure 16: *Bitrate of the scaled and unscaled video stream on a link with 700 kbps available bandwidth*

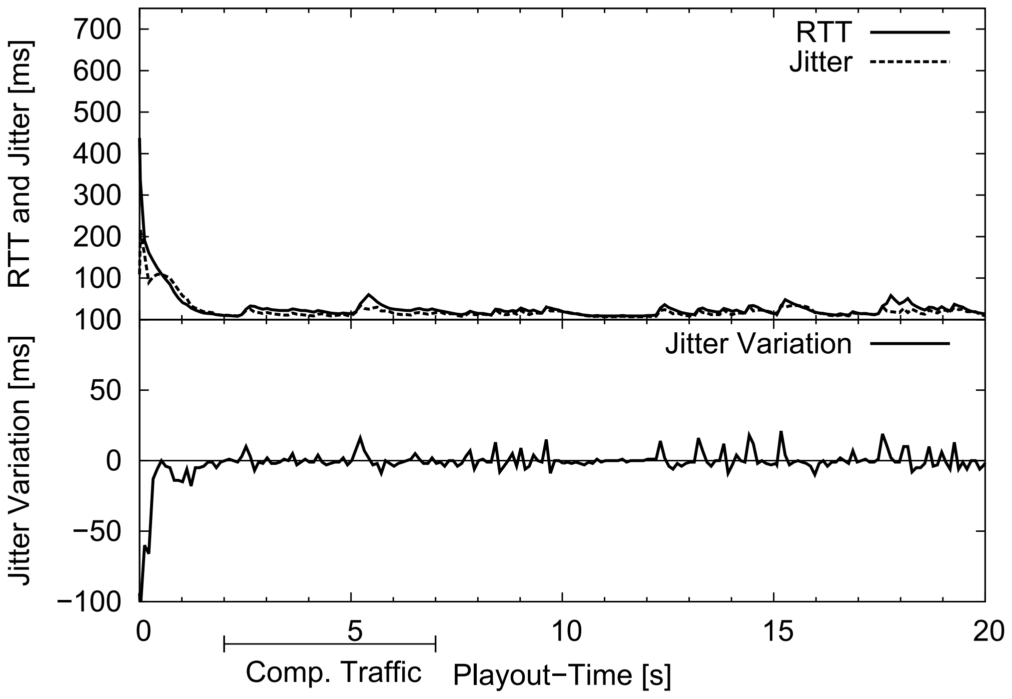

Figure 17: *RTT, Jitter and Jitter Variation on a link with 700 kbps available bandwidth and 100 kbps competing TCP traffic*

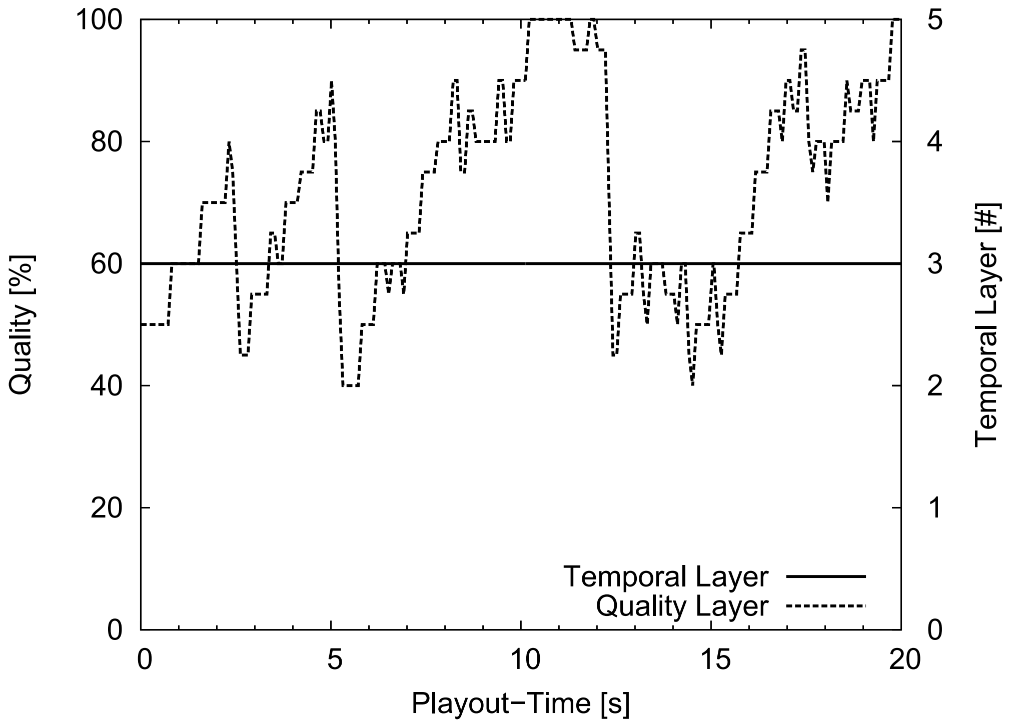

Figure 18: *Quantization and Temporal Layer on a link with 700 kbps available bandwidth and 100 kbps competing TCP traffic*

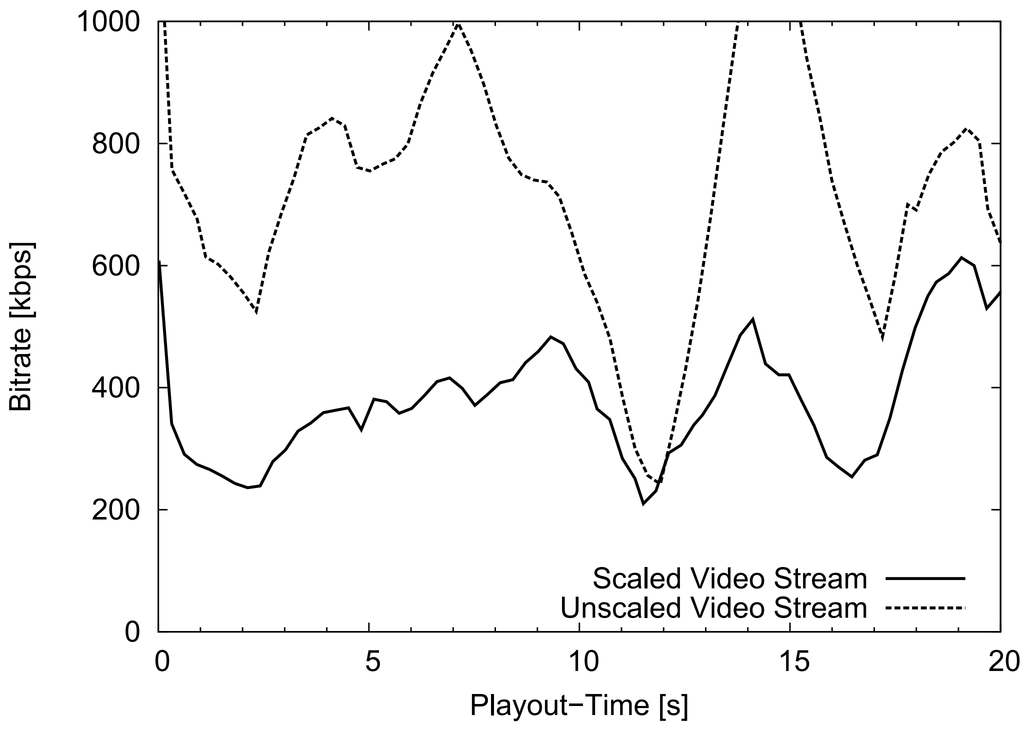

Figure 19: *Bitrate of the scaled and unscaled video stream on a link with 700 kbps available bandwidth and 100 kbps competing TCP traffic*

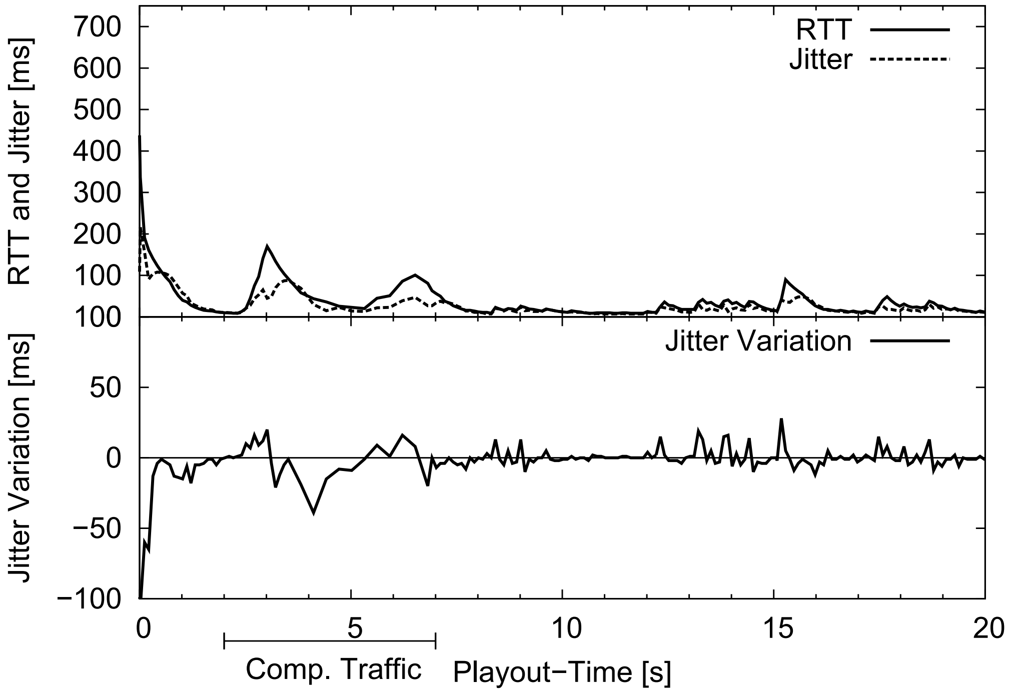

Figure 20: *RTT, Jitter and Jitter Variation on a link with 700 kbps available bandwidth and 100 kbps competing UDP traffic and an adaptive codec*

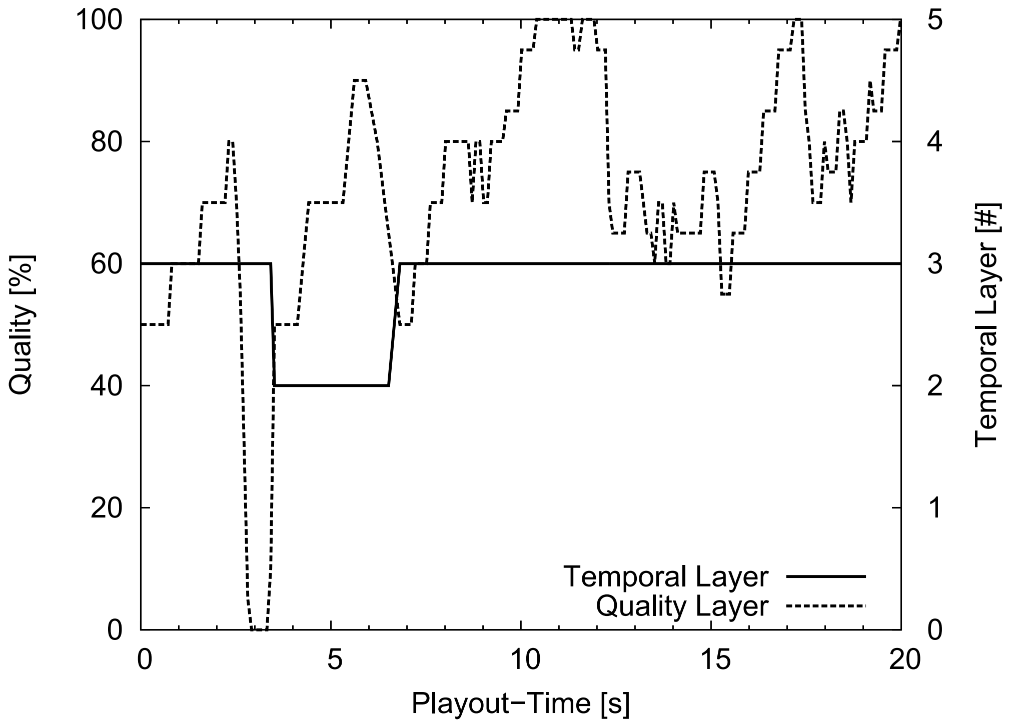

Figure 21: *Quantization and Temporal Layer on a link with 700 kbps available bandwidth and 100 kbps competing UDP traffic*

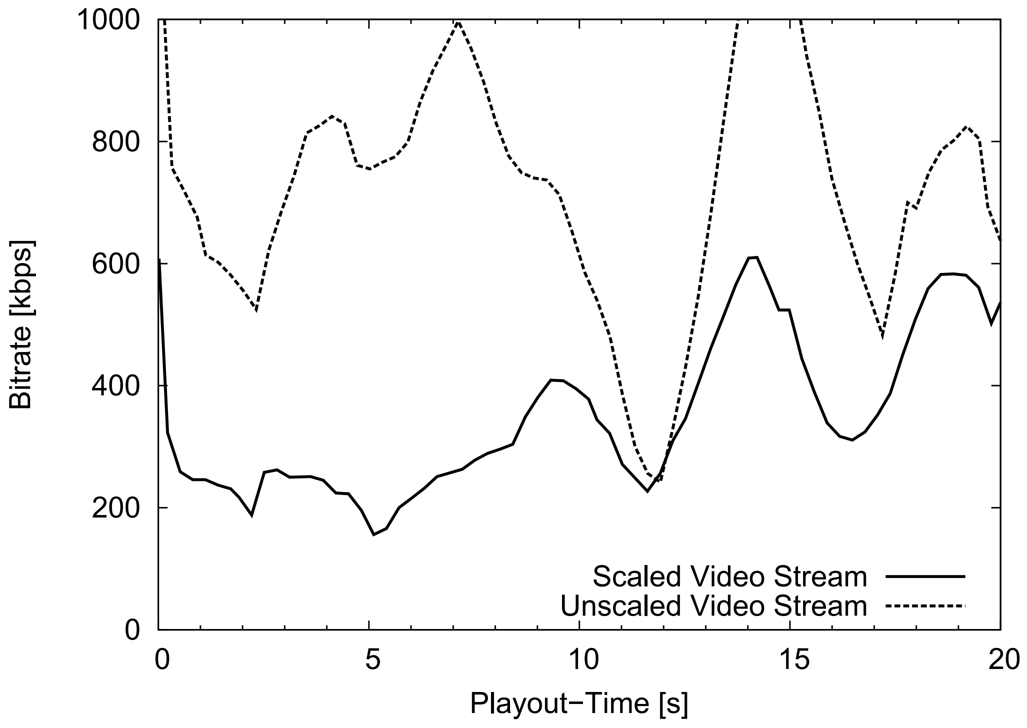

Figure 22: *Bitrate of the scaled and unscaled video stream on a link with 700 kbps available bandwidth and 100 kbps competing UDP traffic*

9.3 Test Results with Competing UDP Traffic

The results for a congested link with 100 kbps UDP competing traffic are shown in Figures 20 and 21. After the competing UDP traffic starts at 2 s, the jitter variation increases heavily and the congestion control reduces the quality to a minimum. Nevertheless, the link will still suffer from congestion, and we also have to remove an additional temporal enhancement layer. Our algorithm removes an enhancement layer if the quality stays below 40% for 8 frames or longer. This applies in this scenario and avoids the congestion successfully. After the 7 s mark (end of competing traffic) the quality stays above 60% for 8 frames or longer, which indicates enough bandwidth to add a temporal enhancement layer. In comparison to the results without the congestion control in Figure 13, the RTT is significantly lower and the link never congests.

In Figure 22 the bitrates of the scaled and the unscaled video stream is shown. Our objective in this scenario is to keep the video bitrate below the 700 kbps to ensure a good QoE for the user. The bitrate of the unscaled video stream often exceeds this limit, while the scaled video stream stays below the 700 kbps most of the time.

10 Conclusion and Outlook

Multimedia applications with high quality video streams need awareness of the network conditions to ensure a high QoE for each participant. We presented a sender-based, fast and lightweight video adaptation scheme, which is capable of scaling the video codec based on the network conditions without a complex and slow approximation of the available bandwidth. Instead, we used the jitter variation as an indicator to predict and avoid a congestion on the link. Since we did not approximate the exact bandwidth, we also had to find a new approach to adapt the video codec. We tried to use the temporal layer and the quantization to scale the codec regardless of the best QoE. In the future, we will also consider the spatial and quality layer for scaling and also factor the QoE into the scaling decision.

We developed an application to ensure that this approach is working properly. The application encodes a raw video, sends it over the network and obtains relevant information about the network on the transport layer. We analyzed the results and used them to scale the video codec for an optimal bandwidth usage. Our test results showed, that this approach ensures a fast and accurate link observation and is capable of recognizing a link congestion early enough to avoid it before it gets noticeable to the user. Especially on links with heavily changing traffic, a quick reaction is important for real-time multimedia applications rather than a slow but more accurate measurement. In contrast to common approaches, the jitter variation observation approach handles this trade-off very well. In the future, we will use PlanetLab [11] to test our approach in a real network.

In conclusion, we experimentally proved that it is possible to recognize and avoid congestion on a link just by observing the RTT, the jitter and the jitter variation without any knowledge about the actual available bandwidth. This approach especially aims for real-time applications that need this information as fast as possible and that are capable of reacting to them.

References

[1] H. L. Cycon, T. C. Schmidt, M. Wählisch, D. Marpe, and M. Winken, “A Temporally Scalable Video Codec and its Applications to a Video Conferencing System with Dynamic Network Adaption for Mobiles,” *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1408–1415, August 2011. [Online]. Available: http://dx.doi.org/10.1109/TCE.2011.6018901

[2] H. L. Cycon, T. C. Schmidt, G. Hege, M. Wählisch, D. Marpe, and M. Palkow, “Peer-to-Peer Videoconferencing with H.264 Software Codec for Mobiles,” in *WoWMoM08 – The 9th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks – Workshop on Mobile Video Delivery (MoViD)*, R. Jain and M. Kumar, Eds., IEEE. Piscataway, NJ, USA: IEEE Press, June 2008, pp. 1–6. [Online]. Available: http://dx.doi.org/10.1109/WOWMOM.2008.4594916

[3] T. Schierl, T. Stockhammer, and T. Wiegand, “Mobile Video Transmission Using Scalable Video Coding,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1204–1217, September 2007.

[4] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007.

[5] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC), “Advanced Video Coding for Generic Audiovisual Services,” ITU, Tech. Rep. 8, July 2007.

[6] G. Urvoy-Keller, T. En-Najjary, and A. Sorniotti, “Operational comparison of available bandwidth estimation tools,” *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 1, pp. 39–42, January 2008.

[7] Lee Salzman, “Enet,” 2012. [Online]. Available: http://enet.bespin.org

[8] M. Palkow, “The daViKo homepage,” 2012, http://www.daviko.com.

[9] “Wondershaper,” 03.03..2012. [Online]. Available: http://lartc.org/wondershaper/

[10] ITU, “G.114 - One-way transmission time,” ITU, Recommendation - Telecommunication Union Standardization Sector, 05 2003.

[11] “The Planet-Lab Central homepage,” http://www.planet-lab.org, 2010.

[12] Q. Liu and J.-N. Hwang, “End-to-end available bandwidth estimation and time measurement adjustment for multimedia QOS,” in *Multimedia and Expo, 2003. ICME ’03. Proceedings. 2003 International Conference on*, vol. 3, 2003, pp. III–373–6 vol.3.

[13] T. Tunali and K. Anar, “Adaptive available bandwidth estimation for internet video streaming,” *Signal Processing: Image Communication*, vol. 21, no. 3, pp. 217–234, March 2006. [Online]. Available: http://dx.doi.org/10.1016/j.image.2005.10.001

[14] I. Richardson, *Video Codec Design: Developing Image and Video Compression Systems*. Wiley, May 2002. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0471485535

[15] H. Schwarz, D. Marpe, and T. Wiegand, “Overview of the Scalable Video Coding Extension of the H.264/AVC Standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, September 2007. [Online]. Available: http://dx.doi.org/10.1109/TCSVT.2007.905532