Scalable Video Coding in Heterogeneous Conference Scenarios

Fabian Jäger

Projektarbeit 2

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Contents

List of Figures

1 Introduction

The data demands of high quality video streams in multimedia applications often exceed the available bandwidth in networks and lead to congestions. A congestion limits the quality and can be avoided with an individually scalable video stream for each participant, which is adapted to the available bandwidth. The adaptation of the video stream to the network conditions is a complicated task, because it is neither easy to identify the available bandwidth nor to determine a congested link. In general, the conditions on a link can be measured at the sender or at the receiver-side and both approaches have advantages and disadvantages.

In this paper, we will focus on a receiver-sided congestion indication. We develop an approach, which obtains information about the network conditions by observing variations of the frame rate, that are caused by traffic on a congested link. Further, we will test and compare the receiver-sided approach to a sender-sided congestion indication and discuss the results.

The paper is organized as following. Chapter 2 discusses the problems in a video conferencing software. Chapter 3 introduces different approaches to determine a congested link. In Chapter 4, we discuss the codec adaptation algorithm. Our testbed and sample application are presented in Chapter 5. Chapter 6 contains the measurement results. Chapter 7 contains a conclusion and an outlook.

2 Problem of Adaptive Video Scaling and Related Work

A challenge in video conferencing applications is to establish a system with multiple participants, bidirectional video streams and different network conditions. To accomplish a stable video stream transmission for all participants, the video codec can be adapted to the available bandwidth. The estimation of the available bandwidth and the scaling of the video stream are the challenges, which are examined in this work.

The bandwidth demands of the video stream mainly depends on the codec in use, the quality and the compression complexity. On 25 January 2013 the new codec standard High Efficiency Video Coding (HEVC, ITU-T H.265 or ISO/IEC 23008-2 [23]) was released, which only needs half of the bitrate of its predecessor ITU-T H.264 [23]. The HEVC can provide a smaller video stream with the same quality as H.264, but also requires more processing power.

In this work, we focus on the H.264 and its SVC extension that have important implications for the network transport. 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 [19],[3] and can be done with the scalable video coding (SVC [21]), which is an extension to H.264 [11]. With SVC, we are able to adapt the video codec to the available bandwidth.

For the video codec adaptation, an estimation of the available bandwidth is needed. It is not easy to find a continuous and accurate measurement of the available bandwidth, because the available bandwidth on a path is not a constant value, but fluctuates and depends on multiple factors like queuing delays, side traffic and heterogeneous link capacities. In general, the available bandwidth can be measured at the sender-side or at the receiver-side and both sides have vision of different network metrics. The sender for example, can obtain the round-trip time (RTT), while the receiver has no information about the RTT, but it is aware of network metrics like the buffer queue at the receiver-side. The sender-sided scaling was already implemented in a previous work [14] and therefore we will focus on the receiver-sided bandwidth estimation in this work to compare both approaches.

The common approaches to measure the available bandwidth like PGM (Probe Gap Model) or PRM (Probe Rate Model) are independent of the application area and require extra probing packets [24]. The accuracy of the bandwidth estimation depends on the amount of probing packets, the nature of side traffic and the duration of the measurement. For further information on these approaches, common bandwidth estimation tools are presented and compared in [7][8]. These techniques are intrusive and rather slow, but we will present an approach in this work, which adapts the PGM approach to a real-time video streaming application. PGM sends probing packets with a small, fixed gap (time of transmission difference between two packets) between them over a path to measure the influence of potential side traffic. On an uncongested path, the gap between the packets is not influenced by any queuing delays and the gap between the two packet is the same at the receiver-side (as long as the bitrate of the probing packets is below the capacity of the path; Otherwise the gap is also influenced by the capacity of the path). If at least one link on the path is congested, the probing packets are also influenced by the competing side-traffic, due to the queuing delays on the router. The gap increases due to the queuing delays and differs from the gap the packets had at the sender-side. With this variations of the gap between the probing packets, the receiver is able to estimate the available bandwidth on path.

On the application layer , the video stream can be used to obtain information about the network conditions. This approach is used in web-based video players like Open Video Player (OVP) [18] or the Strobe Media Playback, which uses the Open Source Media Framework (OSMF) [6]. These video players are specialized to play videos, which are hosted on websites and are delivered via a HTTP stream [16]. The server provides the same video with different bitrates. The streaming client is made aware of the available bitrates by a manifest file and requests one of them. The bandwidth estimation is done at the client side, which monitors the filling level of the video buffer. If the buffer chains drops below a certain threshold, the path has not enough available bandwidth to deliver the video stream in the current quality and the receiver requests a video stream with a lower bitrate. If the buffer is filled over a longer period of time, the client requests a video stream with a higher bitrate. In most scenarios, the video is pre-encoded with different bitrates, but it would also work for a scalable video stream. This approach is often used for HTTP stream based applications, because HTTP is widely deployed and the content providers mostly already have a HTTP infrastructure. Non-HTTP streaming solutions would require an additional specialized streaming server infrastructure. On the downside, this approach is more complex than other streaming technologies. The video stream is transmitted via TCP [13], which simplifies the traversal of firewalls and NATs, but it already has a congestion control and might inflict the application layer congestion control [1]. Also, a buffer-based video delivery estimation is not usable in a real-time video application like a video conferencing software, because the buffering at the receiver-side increases delay in an undesirable fashion.

Another approach to obtain information about the network conditions is to examine the video stream directly with a PGM approach. D. T. Nguyen and J. Ostermann present such a scheme to scale a video stream at the receiver-side [15]. They present a streaming system, which uses the scalability extension of H.264/AVC and provide a congestion control algorithm. The system is tested with UDP and TCP as competing traffic and the results showed, that the system reaches a throughput which is at least 50% higher than existing congestion control algorithms. Their approach is similar to PTR, which is a PGM approach [9]. PTR uses at least three sets of 60 packets with 700 bytes for one bandwidth estimation. This is significant overhead, especially for a video streaming application, where a regularly performed bandwidth estimation is needed. Instead of extra probing packages, the sending time of the RTP [20] video packets is manipulated to use them also as probing packets. Therefore, the RTP packets are sent with a certain gap between them. Like in PGM, the competing traffic will influence the gap between the RTP packets. On the receiver-side, the gap between the RTP packets is measured and used in a modified bandwidth estimation formula from the tool PTR. The modification is necessary, because in PTR the size of the packets are constant, while the size of the RTP packet varies. In a real-time streaming application, the modification of the RTP sending time behavior is a problem, because it is important to send the video frames as fast as possible without any delays. Altogether, this approach gives a good overall insight to the receiver side bandwidth estimation, but it is not optimized for a real-time multimedia application.

In this work, an approach is used, which is inspired by the PGM approach. Instead of extra probing packets, the gap between the video frames is used to gather information about the network conditions. This reduces the overhead and is also independent of the transport protocol.

3 Congestion Indication

The challenge for the congestion indication in a multimedia application with a real-time video stream is to determine the network conditions as fast as possible and to react properly by scaling the video stream to avoid congestions on the path. A fast reaction has the priority, and the accuracy of the measurement is secondary.

We have developed an receiver-sided approach to detect a congestion on a link and take immediate action to avoid it. However, a congestion can also be indicated at the sender-side, which was presented in our earlier work [14]. In the following, we give a short introduction to both approaches and discuss their advantages and disadvantages.

3.1 Sender-Sided Congestion Indication

The video stream is encoded at the sender-side and the sender has a direct influence on the scaling. Therefore, a sender-sided video adaptation is faster than a receiver-sided video adaptation, which needs an additional response channel to inform the sender about the codec scaling. The network metrics visible at the sender-side depend on the protocol, which is used to transmit the video stream. Most protocols have in common that they usually provide information about the transmission time (usually only the RTT and not the one-way delay). The challenge is to analyze the provided network metrics to detect a congested link as fast as possible and approximate a proper codec scaling.

The general idea for a fast sender-sided video adaptation is to obtain indirect information about the available bandwidth by measuring the one-way delay on the transport layer and analyzing its jitter [3]. Every congestion adds a significant queuing delay to the one-way delay which differ from the regular one-way delay fluctuations. However, the one-way delay is not easy to measure and requires an approach with synchronized clocks [4]. This is complex and also produces overhead, especially in large multimedia applications with multiple participants. Instead of the one-way delay, the round-trip time (RTT) is used to determine the queuing delays, which is a more light-weight method but may also fail when the queuing delay occur on the return path and not on the forward path, which is used to transmit the video stream.

The RTT itself cannot be used for a fast congestion detection, since the RTT mainly depends on the topology and a high RTT does not necessarily indicate a link congestion. Instead, we are trying to detect a congestion by a rising RTT, which differs from the regular RTT fluctuation. Therefore, the jitter and its variation (which is the second derivative of the RTT) is used to determine a congestion, which is identified by a significant and discontinuous jump in the jitter variation that goes along with the congestion.

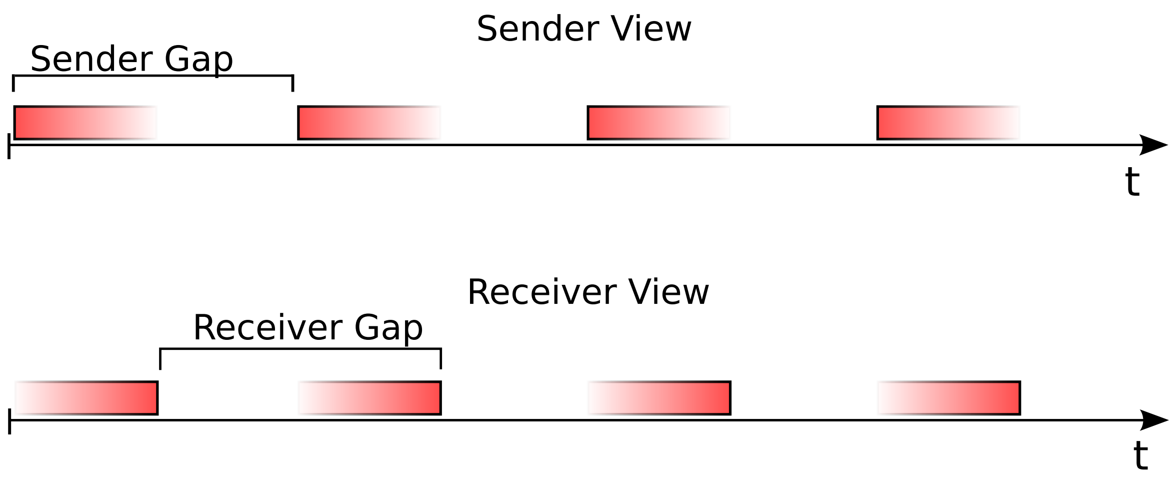


***Figure 1:** Sender and receiver views on the gap between two video frames*

In Section 4.3, we discuss this approach and optimize the video stream adaptation.

3.2 Receiver-Sided Congestion Indication

The receiver has no direct influence on the transmission of the video flow, but can request the sender to scale the video stream. The metrics the receiver can obtain from the received video frames are values like the size of the frame, the filling level of the receive buffer and the time of reception. In comparison with the sender-sided approach, which uses packets to measure the network conditions, the receiver-sided approach uses complete frames. The challenge is to find an approach to determine a congestion based on the given metrics and request a scaled video stream accordingly.

The general idea at the receiver-side is to use the framerate to detect changing network conditions. If the framerate on the receiver-side stays below the original framerate of the sender-side, a link might be congested. Instead of measuring the framerate over a period of time, we use the interarrival gap between two frames to predict a changing framerate.

The interarrival gap at the sender and the receiver is shown in Figure 1. At the sender-side, the gap between the frames is measured when the frames are sent and is ideally the inversion of the framerate ($\Delta s = \frac{1}{framerate}$). The framerate at the receiver-side is approximated with the time of frame reception. At the receiver-side, the gap between two frames is

$$\Delta r = rx_i - rx_{i-1} \tag{1}$$

where rx_i is the time of receiving the last bit of the i-th frame. The measured gap Δr is the original gap Δs with additional queuing delay variation Δq at the router and retransmission delay variation Δd (when supported by the protocol):

$$\Delta r = \Delta s + \Delta q + \Delta d \tag{2}$$

In this work, we will focus on the queuing delay Δq and for this purpose, we assume no packet dropping occurs and Δd to be zero.

On a congested link, the frame transmission time is stretched due to increasing queuing delays at the router. The stretched frame transmission time increases the gap between the frames at the receiver side and the playout time is delayed. Analyzing Δr is comparable to the PGM approach, but instead of packets, video frames are used for the measurement.

The receiver is comparing the gap between two frames Δr with the gap Δs they had when the sender sent the two frames. Therefore, the receiver needs also information about the gap between the frames at the sender-side. For a video stream with a constant frame rate, the gap at the sender-side is always constant ($\Delta s = \frac{1}{framerate}$). In this case, the receiver only needs to know the frame rate of the video. If the framerate varies, the information about the gap at sender-side needs to be transmitted to the receiver.

If no queuing delay changes occur ($\Delta q = 0$), the gap at the sender-side and the gap at the receiver-side ideally do not differ (for simplification we assume that every video frame has the same size). If queuing delays rise, the frames do not arrive in time and the gap at the receiver-side is bigger than the sender-sided gap ($\Delta r > \Delta s$).

In Figure 2 this approach is visualized. Views of a video stream from the sender and the receiver are shown for an uncongested and congested link. The first two figures show two frames on a free link and the gap between them. The sender only knows the time of transmission for the frames and the receiver knows only the time of reception. None of them does know the transmission time for a frame. On the congested link, the transmission time for the frames increases due to the queuing delays. This is not visible at the sender-side and the gap is still the same here. At the receiver-side, the gap increases and differs from the gap at the sender-side. With the sender and receiver frame gap difference, the receiver is able to detect this congestion.

In conclusion, as long as the bitrate of the video stream is lower than the available bandwidth, the gap at the receiver-side stays constant and is equal to the sender gap. However, if the bitrate of the video stream is higher than the available bandwidth, the packets get queued at the router and the gap at the receiver-side increases and differs the sender gap significantly (50ms - 150ms difference).

The specific implementation of this approach is described in 4.2.

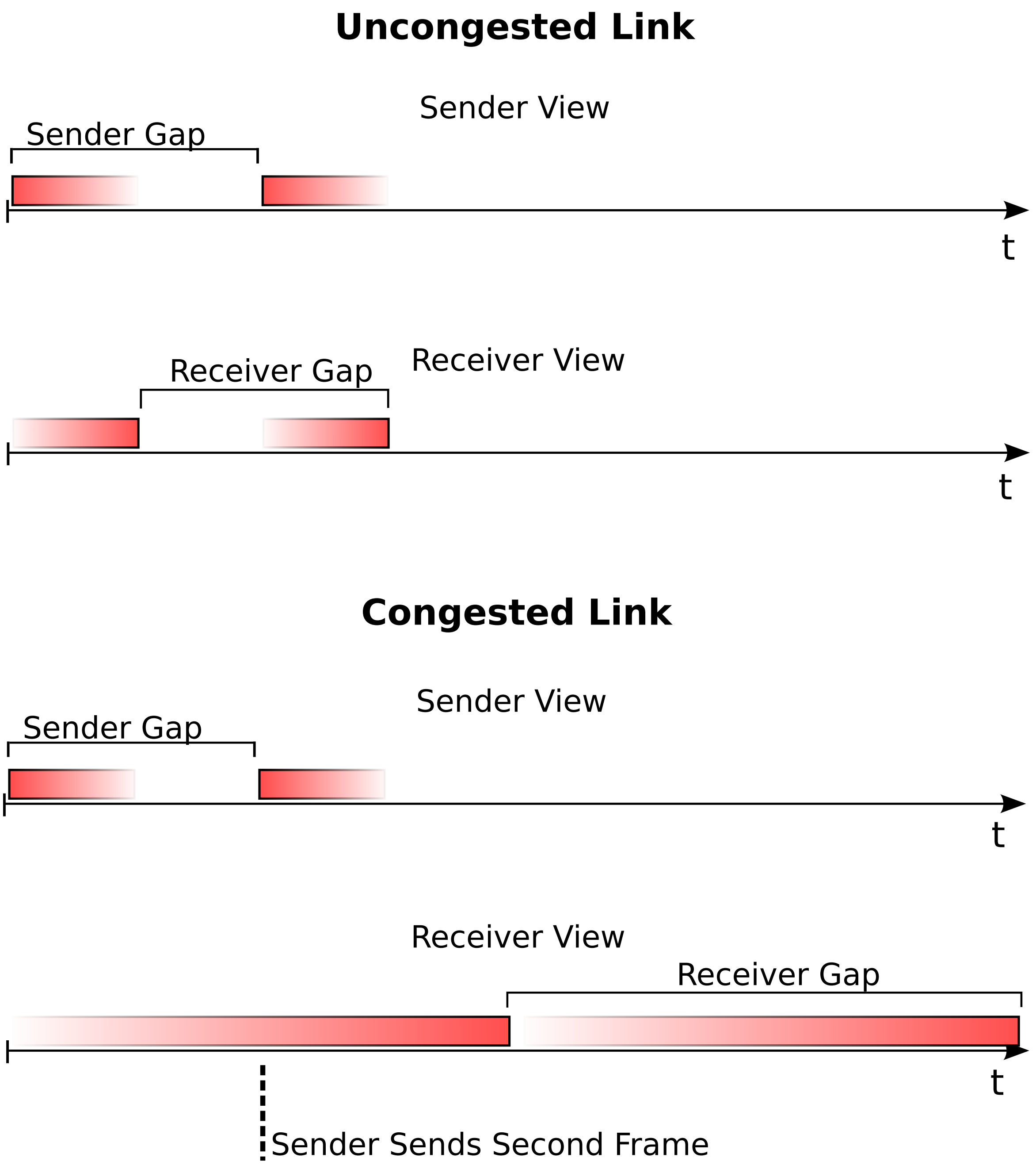


Figure 2: *Sender gap and receiver gap between two video frames on an uncongested and on a congested link*

3.3 Comparison between Sender and Receiver Congestion Indication

The congestion indication at the sender-side predicts a possible congestion and scales the quality of the video stream down to prevent the congestion. The sender uses the jitter variation of the RTT as indicator for a congestion prediction, which is a very fast and light-weight approach, but it has also disadvantages. If a link starts to congest, and we do not react appropriately, the path congests and the RTT starts to vary depending on the queuing delays at the router. The problem is, that the sender is able to detect a rising RTT and predict a congestion, but on a congested path, the sender loses the reference to the regular RTT fluctuations and has no way to know that it operates on a congested path. The sender still scales the quality of the video stream in both directions according to the jitter variation, even though the path is still congested. The sender-sided congestion indication has the disadvantage, that if we want to scale the video stream with the jitter variation as indicator, it is necessary that we have a free path at the beginning and prevent all congestions. The advantage of this approach is, that we are able to indicate a possible congestion very fast and can react to it before the path congests.

At the receiver-side, the gap between video frames is used to detect a congestion. The downside of the sender-sided approach is its unreliability on a congested link, because it loses the reference to a free link. In comparison to the sender-sided approach, the receiver-side approach does not search significant jumps in the jitter variation (2nd derivative of the RTT) to detect a congested link. Instead, the additional queuing delay to the frames is measured at the receiver-side. In comparison with the packet based sender-sided congestion control, the measuring period is a whole frame, which is longer and allows a better approximation of the conditions on the link. Therefore, the receiver-sided congestion control is slower, but more accurate.

In an application with both approaches collaborating, a balance of the sender-side and the receiver-side congestion control is necessary to gain the advantages of both.

4 Video Codec Adaptation

The codec can be scaled with the quantization factor and temporal enhancement layer. In this work, we use these two scaling options to implement a two-level scaling. We use the quantization factor for a fine-grained scaling. The temporal enhancement layer is used for a rough scaling, which follows the fine-grained scaling. First, the fine-grained scaling with the quantization factor will be presented.

$RTT(t)$	Round Trip Time
$\beta(t)$	Current bitrate at the video stream
$\beta_{min}(t)$	Bitrate with the lowest quality
$\beta_{max}(t)$	Bitrate with the highest quality
$Q(t)$	Level of quantization for the video stream (0 - 20)
$k(t)$	Scaling suggestion from the congestion control
tx_i	Transmission point in time of frame i
rx_i	Receiving point in time of frame i
Δs	Gap between frames at the sender-side ($tx_i - tx_{i-1}$)
Δr	Gap between frames at the receiver-side ($rx_i - rx_{i-1}$)
$\Delta q(t_i, t_{i-1})$	Additional queuing delay in the interval t_{i-1} and t_i
$\mu(t)$	Available bandwidth
$f(t)$	Filling level of the router queue

Table 1: *Parameters for the video stream adaptation*

4.1 Scaling the Codec

In the application, the quantization factor can be scaled from 20 to 40 (where 40 is the worst quality and 20 is the best quality). For simplification, we will refer to it as level of quantization with 0 as worst quality and 20 as best quality.

The challenge is how to feedback the measurements from the congestion control into the codec adaptation. The suggestion from the congestion control is a relative value k, which is the ratio of video traffic and available bandwidth:

$$k(t) = \frac{\mu(t)}{\beta(t)} \tag{3}$$

This suggestion needs to be mapped to the parameters the codec provides for scaling. In our application, the codec compression and the resulting bitrate can only be changed with the quantization factor and the temporal layer. The codec also scales only in a certain bit range depending on the level of quantization (0-20) and temporal layer (1-3). An example is shown in Figure 3. The graph displayed the measured bit range of our test video with the highest quality (max. bitrate) and with the lowest quality (min. bitrate). The video source can only scale the video stream between these two extremes.

Since the codec only operates in a certain bitrate range, a mapping between the network bandwidth reduction and the quality is needed. In this work, the network conditions are mapped to the quantization factor and the approximated bitrate β for a given quality of quantization $Q(t)$ is calculated as follows:

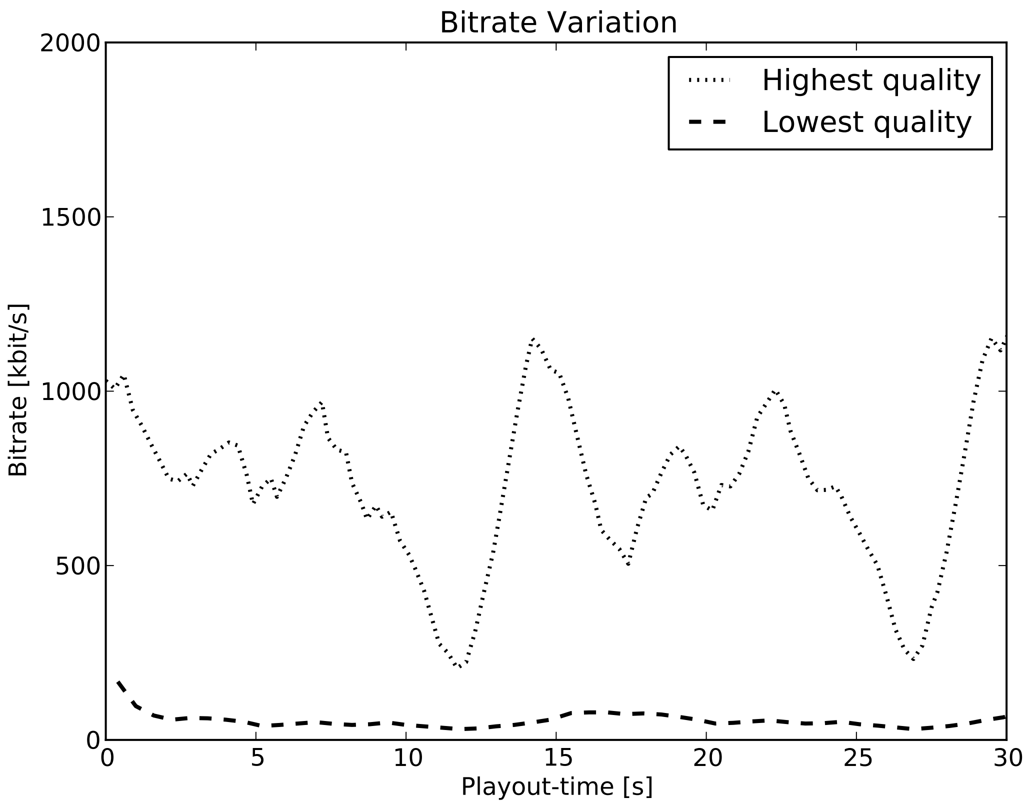


Figure 3: *Bitrate variation of a video stream*

$$\beta(t) = \beta_{min}(t) + \frac{Q(t)}{Q_{max}} * (\beta_{max}(t) - \beta_{min}(t)) \tag{4}$$

Correspondingly, the level of quantization $Q(t)$ for a requested bitrate β can be calculated as following:

$$Q(t) = \frac{(\beta(t) - \beta_{min}(t)) * Q_{max}}{\beta_{max}(t) - \beta_{min}(t))} \tag{5}$$

For example, let β_{max}=100 Mbit/s, β_{min}=50 MBit/s and only the quantization factor is used for scaling. In this scenario, the mapping between the bitrate and the quality of quantization looks as following:

Bitrate of the video [Mbit/s]	Quality of quantization [#]
100	20
90	16
80	12
70	8
60	4
50	0
40	-
30	-
20	-
10	-
0	-

Based on the observed network condition, the congestion control suggests a relative scaling k for the current video stream. The scaling factor k is the ratio from the approximated available bandwidth μ to the current bitrate of the video stream β:

$$k(t) = \frac{\mu(t)}{\beta(t)} \tag{6}$$

If k is below 1, the bitrate exceeds the available bandwidth and the video stream needs to be downscaled by the factor k to match the available bandwidth. Suppose our video at time $t - \delta$ runs at bitrate $\beta(t - \delta)$ and gets rescaled such that $\beta(t) = \beta(t - \delta) * k(t)$. The old bitrate of the video stream $\beta(t - \delta)$ can either be measured or approximated with the old quality of quantization. The new quality of quantization can then be calculated for the current video stream bitrate as follows:

$$Q(t) = \frac{(\beta(t-\delta) * k(t) - \beta_{min}(t)) * Q_{max}}{\beta_{max}(t) - \beta_{min}(t))} \tag{7}$$

In our example, the video stream has the maximum quality (100 Mbit/s) on a link with 100 Mbit/s available bandwidth. If the available bandwidth drops from 100 Mbit/s to 90 Mbit/s, the congestion control will suggest a 10% bitrate reduction.

$$k = \frac{90Mbit/s}{100Mbit/s} = 0.9 \tag{8}$$

The new quality of quantization is calculated with $\beta(t-\delta) * 0.9$ as requested bitrate:

$$Q(t) = \frac{(\beta(t-\delta) * k(t) - \beta_{min}(t)) * Q_{max}}{\beta_{max}(t) - \beta_{min}(t))} \tag{9}$$

$$Q(t) = \frac{(100Mbit/s * 0.9 - 50Mbit/s) * 20}{100Mbit/s - 50Mbit/s} \tag{10}$$

$$Q(t) = 16 \tag{11}$$

The new level of quantization is 16, which approximately generates a 90 Mbit/s video stream, which matches the available bandwidth.

Unfortunately, β_{min} and β_{max} are unknown, because they rely on the individual video. The bitrate of a video stream for a certain quantization factor varies, but for scaling the codec we need at least a rough estimation of the data demands for the different quality settings of the video stream. Several complex approaches exist to estimate the bitrate of a video stream [22], but for our purpose it is sufficient to use a simple exponential moving average for every frame with the highest or lowest level of quality.

4.2 Receiver-side Video Codec Adaptation

The video stream shall be adapted to the condition of the network. At the receiver-side, the inter-frame gap difference between sender and receiver are measured to adapt the video stream. With the measured inter-frame gap difference, we determine the ratio of available bandwidth to the bitrate of the video stream.

On an ideal, uncongested link, the sender gap and the receiver gap are equal. If the link is congested, the queuing delays are added to the gap at the receiver-side. The gap at the receiver-side Δr is the sum of Δs and an additional queuing delay Δq, which is responsible for a the increased inter-frame gap. The queuing occurs at routers that have an egress to a congested link. If we examine a single video stream, the queue fill level variation in the measurement interval t_{i-1}, t_i can be calculated with the video bitrate β and the available bandwidth μ:

$$f(t_i) - f(t_{i-1}) = \int\limits_{t_{i-1}}^{t_i} \beta(\tau) - \mu(\tau)\, d\tau \tag{12}$$

The expected queuing delay q can be calculated by the change of the filling level and the departure rate of the queue, which we assume to be equal to the available bandwidth. The additional queuing delay Δq in the interval t_{i-1}, t_i can be approximated under this conditions as:

$$\begin{align} \Delta q(t_i, t_{i-1}) &\approx \frac{f(t_i) - f(t_{i-1})}{\mu(t_i)} \tag{13} \\ &\approx \frac{1}{\mu(t_i)} \int\limits_{t_{i-1}}^{t_i} (\beta(\tau) - \mu(\tau))\, d\tau \tag{14} \end{align}$$

For a small interval, we assume the bitrate and the available bandwidth constant $\mu(t) = \mu,\ \beta(t) = \beta$:

$$\begin{align} &\approx \frac{(\beta - \mu) * (t_i - t_{i-1})}{\mu} \tag{15} \\ &\approx (\frac{\beta}{\mu} - 1) * (t_i - t_{i-1}) \tag{16} \end{align}$$

With this approximation, the ratio k of the available bandwidth μ and the bitrate of the video stream β can also be determined by the ratio of Δr and Δs. The measuring interval is the time between two frames on the sender-side (Δs).

$$\begin{align} \Delta r &= \Delta s + \Delta q(t, t - \Delta s) \tag{17} \\ &= \Delta s + (\frac{\beta}{\mu} - 1) * \Delta s \tag{18} \\ \frac{\Delta s}{\Delta r} &= \frac{\mu}{\beta} = k \tag{19} \end{align}$$

Therefore, the scaling factor k at the receiver-side can be determined by Δs and Δr:

$$\boxed{k = \frac{\Delta s}{\Delta r}} \tag{20}$$

Unfortunately, the gaps between the frames also depend on the frame sizes. The receiver measures the receiving point in time when the whole frame is received and if the second frame is bigger than the first one, the gap increases without any queuing delays. The receiver has knowledge about the frame size, but this does not help, since it has no information about the available bandwidth nor does it know the transmission time for a frame or a packet. A simple but imprecise solution is to eliminate the gap fluctuations by determining the average gap over the previous frames and use this for the quality suggestion calculation. A more accurate solution is to also measure the reception of the first bit to determine the transmission time for the frames. With knowledge of the transmission time, the frame size variation can be eliminated in the equation. In the current state, the first solution is used, but in the future, the second solution will be examined and implemented.

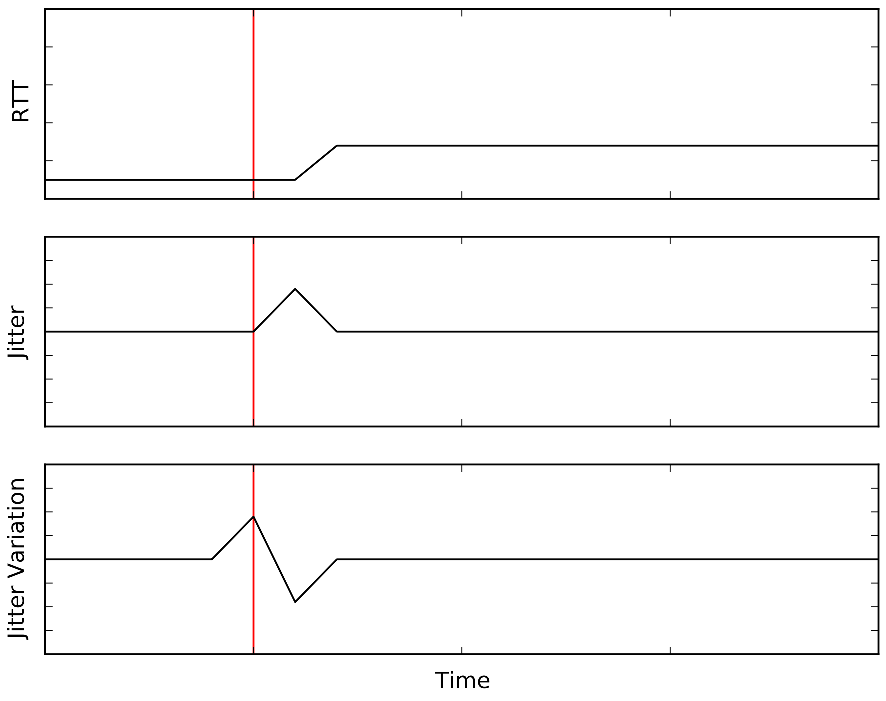
4.3 Optimization of the Sender-Sided Video Adaptation

In an earlier approach to the video adaptation, the jitter variation was compared to the average jitter variation and the quality adjustment was done based on experimentally derived values, which were unrelated to the video stream or the available bandwidth. In this work, the quality decrease is now calculated relative to the jitter variation ratio.

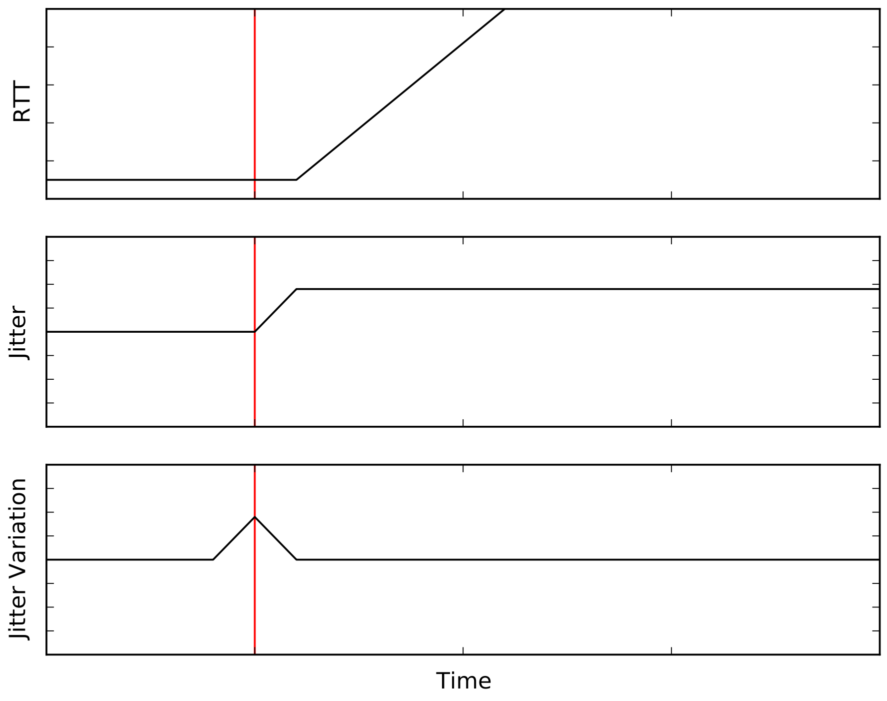
On a congested link, the RTT, jitter and jitter variation start to rise at increasing queuing delays. This increasing queuing delays goes along with a significant jump in the jitter variation (2nd derivative of the RTT), which we use as indicator for a congestion. If the jitter variation exceeds a certain threshold, we assume an impending congestion and scale the video stream down.

If an impending congestion is predicted, an estimation of the the ratio k from the bitrate of the video stream to the available bandwidth is necessary to scale the video stream accordingly. For this purpose, a rough approximated RTT trend for the following packets of the frame is needed. Three base scenarios are possible, which are shown in Figure 4.

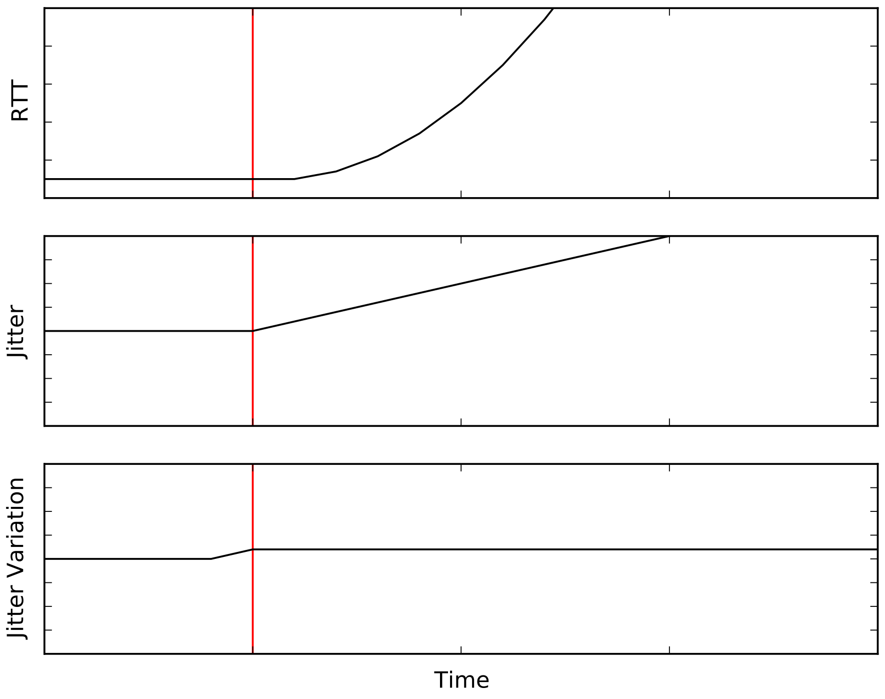
Scenario 1 Assuming the jitter variation is negative, the delay for the following packages would stay the same or get smaller. The RTT would stay the same or show a negative drift.



(a) Scenario 1: Jitter variation < 0



(b) Scenario 2: Jitter variation = 0



(c) Scenario 3: Constant jitter variation

Figure 4: *Predictions for the indicated congestion (red line)*

Scenario 2 Assuming the jitter variation will fall back to zero, the jitter would be constant and the RTT would show a linear increasing behavior. This occurs if the link congests with a constant rate.

Scenario 3 Assuming the jitter variation stay at its current value, the jitter would linearly increase and the RTT would show a exponential increase. This occurs if link congests with an increasing rate.

The challenge is to predict the right scenario after a jump in the jitter variation is detected. In this work, we assume Scenario 2 to be the best option, because Scenario 3 is a rather unusual worst case scenario where a link congests with a growing rate. We prefer Scenario 2 over Scenario 1, because an overreaction in some situations is more desirable than an congested link.

Scenario 2 assumes that the jitter variation will fall back to zero after a short time and the congestion shows a linear growing RTT. Therefore we assume RTT'' to be zero. In general, the RTT increase for n packets is:

$$\Delta RTT(x_n, x_1) = RTT(x_n) - RTT(x_1) \tag{21}$$

$$= \sum_{i=1}^{n-1} (RTT(x_{i+1}) - RTT(x_i)) \tag{22}$$

For Scenario 2, with a constant linear growing RTT and the assumption that:

$$RTT(x_{i+1}) - RTT(x_i) \approx RTT(x_2) - RTT(x_1) \tag{23}$$

the RTT trend can be calculated as:

$$\Delta RTT(x_n, x_1) = \sum_{i=1}^{n-1} (RTT(x_2) - RTT(x_1)) \tag{24}$$

$$= n * (RTT(x_2) - RTT(x_1)) \tag{25}$$

For the queuing delay Δq, the one-way delay is needed instead of the RTT:

$$\Delta q(x_n, x_1) = n * \left(\frac{RTT(x_2)}{2} - \frac{RTT(x_1)}{2} \right) \tag{26}$$

$$= \frac{n}{2} * (RTT(x_2) - RTT(x_1)) \tag{27}$$

In a small measurement interval x_2, x_1, the the available bandwidth μ and the bitrate of the video stream β do not fluctuate much and are assumed to be constant. As a rough approximation, Δq can also be determined with the Formula 16:

$$\Delta q(x_n, x_1) \approx (\frac{\beta}{\mu} - 1) * n * (x_2 - x_1) \tag{28}$$

With these two formulas, where μ and β are unknown, the ratio k from β to μ can be calculated:

$$(\frac{\beta}{\mu} - 1) * n * (x_2 - x_1) = \frac{n}{2} * (RTT(x_2) - RTT(x_1)) \tag{29}$$

$$\frac{\mu}{\beta} = \frac{(x_2 - x_1)}{\frac{1}{2} * (RTT(x_2) - RTT(x_1)) + (x_2 - x_1)} \tag{30}$$

$$\boxed{k = \frac{(x_2 - x_1)}{\frac{1}{2} * (RTT(x_2) - RTT(x_1)) + (x_2 - x_1)}} \tag{31}$$

This is just a prediction for k and the conditions in the near future may differ from this approximation, but this approach allows a very fast reaction to changing network conditions. If the additional queuing delay $\Delta q(x_n, x_1)$ exceeds a certain threshold[1], the frame is delayed at the receiver-side and the video stream needs to be downscaled.

This approach allows a very fast and light-weight codec scaling, which predicts congestions at its beginning. This is a advantage over the receiver-sided approach. On the other hand, the sender just estimates the effects of the queuing to the video stream, which can be less accurate in some situations.

4.4 Combining the Sender and Receiver Adaptation

Both, the sender-sided and the receiver-sided approaches, operate independently and ensure a reliable video adaptation. Both have advantages and disadvantages. Most of the time both approach will provide slightly different quality suggestions and the sender must decide which to use.

4.4.1 Scaling the Quality Layer of the Video Stream

The sender gets quality scaling recommendations from the receiver k_r and from his own measurements k_s. For the video adaptation an Additive Increase/Multiple Decrease (AIMD)

[1] In this work, 25ms is used as threshold.

approach is used [12]. If a quality increase is suggested ($k > 1$), the quality is increased by small additive steps instead of a scaling with the factor k, while the quality reduction is a multiple decrease with the factor k.

Condition	Adaptation
$k_s(t), k_r(t) \geq 1$	Conservative and stepwise quality increase
$k_r(t) < 1 \leq k_s(t)$	Downscaling with $k_r(t)$
$k_s(t) < 1 \leq k_r(t)$	Downscaling with $k_s(t)$
$k_s(t), k_r(t) < 1$	Downscaling with $k_r(t)$

Table 2: *Quality scaling decision*

Table 2 shows how the video adaptation reacts to the recommendation from the sender and the receiver. If both recommend a quality increase, the quality is conservatively increased by a small step. If one of recommends a downscaling ($k < 1$) while the other recommends an upscaling or no scaling ($k \geq 1$), the downscaling is always preferred over an upscaling or no scaling. The quality recommendation from the receiver is taken if both recommend a downscaling. The receiver is preferred, because its measurement results are more accurate and reliable.

4.4.2 Scaling the Temporal Layer

The number of temporal layers is not directly influenced by the bandwidth estimation, but instead depends on the fluctuation of the quality adaptation. 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 values are experimentally determined and do not scale at the current state. This strategy is based on the quality and is only indirectly influenced by the jitter variation or the gap 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.

5 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 use a small network, which is shown in Figure 5 to test our streaming application. It consists of a sender, a receiver and a router connecting

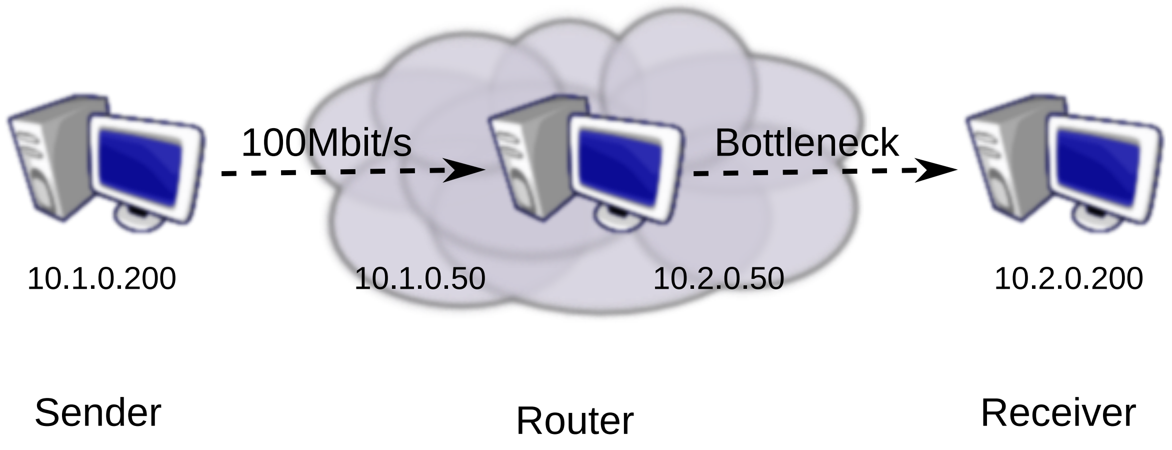


Figure 5: *Testbed*

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, 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.

5.1 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 [25], 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 add network delays and also limit the capacity of the up- and downstream for each interfaces.

5.2 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.

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 stream. The applications might also react to congestion and change their traffic depending on the available bandwidth. This variety is complex to simulate and in work we will focus on the available bandwidth and the delay of a path.

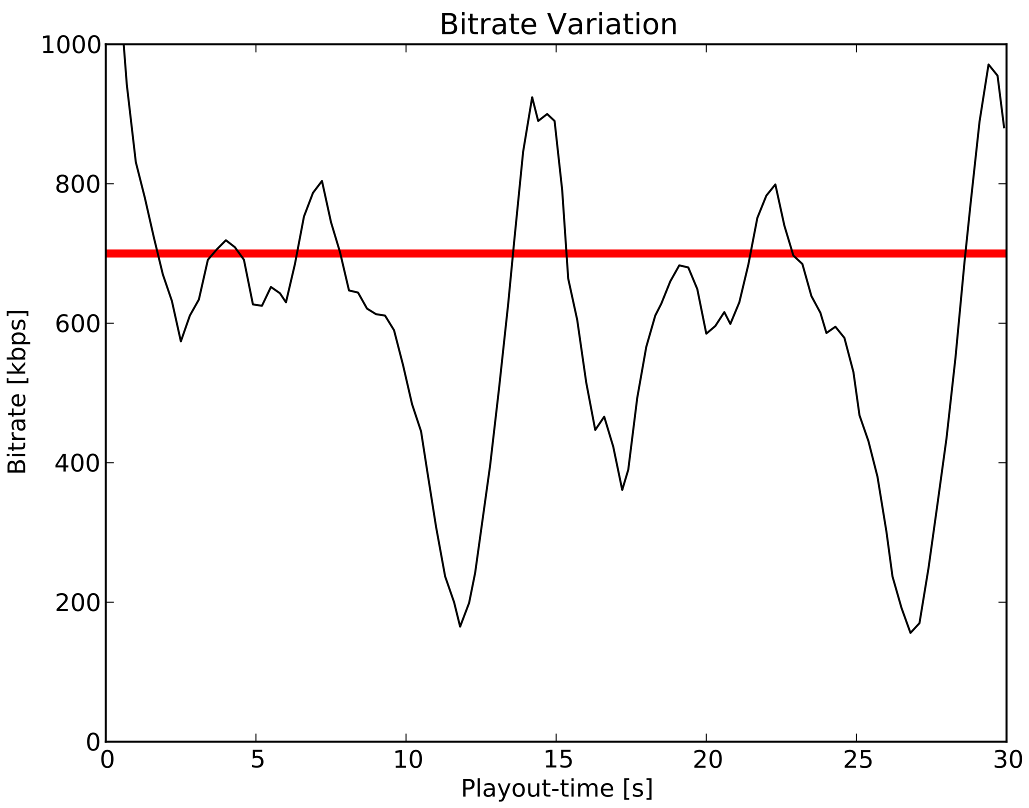


Figure 6: *Bitrate variation of the video stream with 700 kbps available bandwidth*

6 Testscenarios

For testing, we used a 30 seconds 768x576 pixel video with 10 fps as test sequence. At the beginning and after 15 seconds an intra-frame is encoded, which is significantly bigger than an inter-frame. The video frames are transmitted via the enet protocol, which operates on top of UDP and provides an optional, in-order delivery of packets [5]. Since the enet protocol initializes the RTT and jitter with 500 ms, it takes a short time until we have usable values.

The bandwidth characteristic of the video is shown in Figure 6. The x-axis represents the time in units of seconds, while the y-axis pictures the approximated average bitrate over time for a video stream with highest quality. As we can see, the video bitrate varies between 200 and 1000 kbps and it is possible to transmit the video on a path with 1000 kbps available bandwidth without congesting the link. In this work, we will set the available bandwidth to 700 kbps, which is not enough for some parts of the video stream. Especially at the 15s mark, the bitrate of the video stream exceeds the available bandwidth clearly.

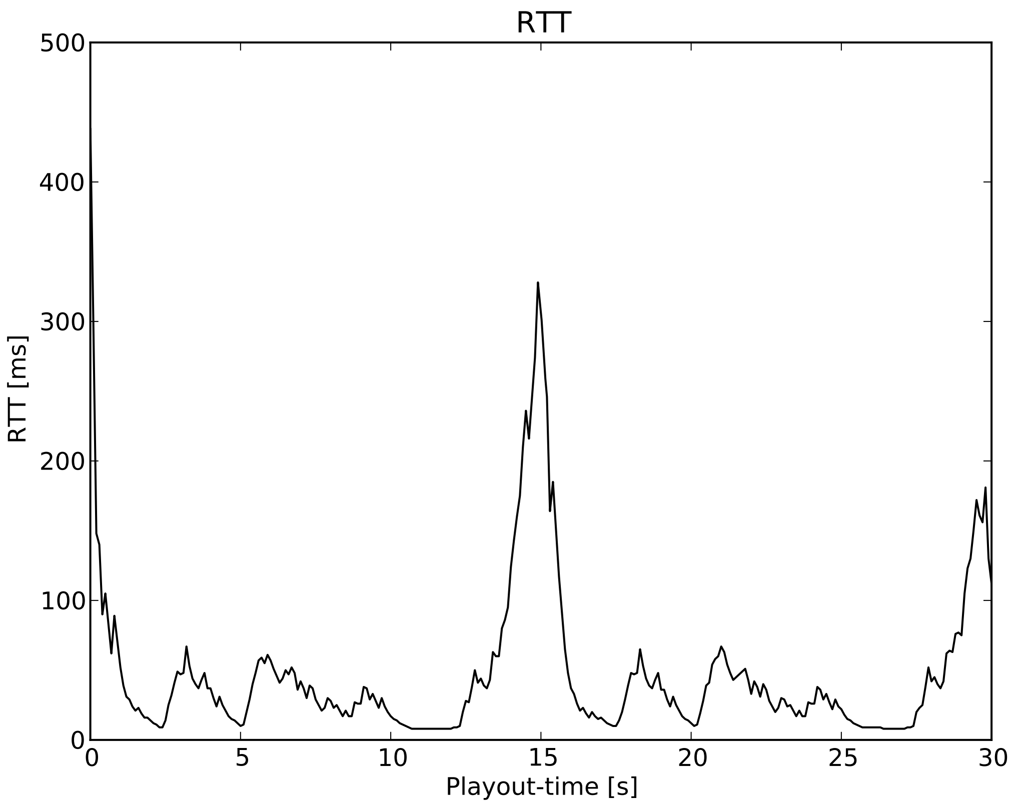


Figure 7: *RTT on a path with 700 kbps available bandwidth*

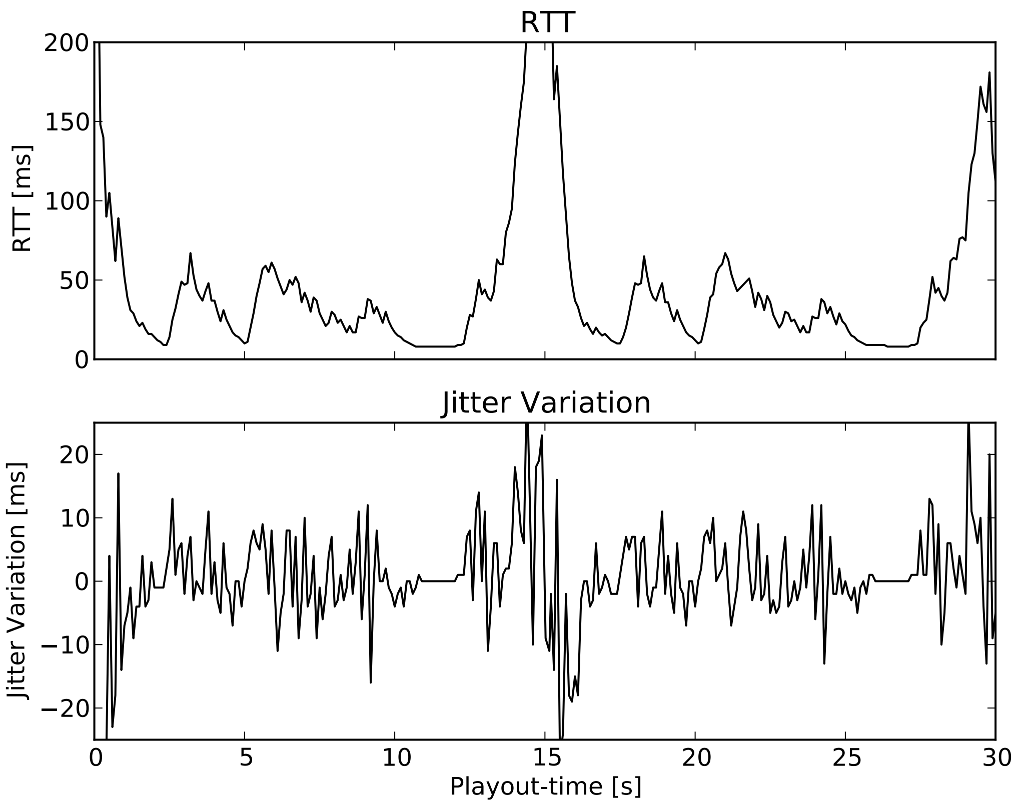


***Figure 8:** RTT and Jitter Variation of an unscaled video stream*

This congestions are visible in the RTT, which is shown in Figure 7. 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].

6.1 Sender-Sided Video Scaling

The sender uses the jitter variation as an indicator for changing network conditions. In this scenario, only the sender scales the video codec without consideration of the receiver.

Figure 8 shows the RTT and the jitter variation of an unscaled video stream on a link with 700 kbps available bandwidth. The RTT and the jitter variation are both unstable with huge fluctuations, especially at the 15s mark, where a new intra-frame is encoded.

Figure 9 shows the RTT of a sender scaled video stream on a link with 700 kbps available

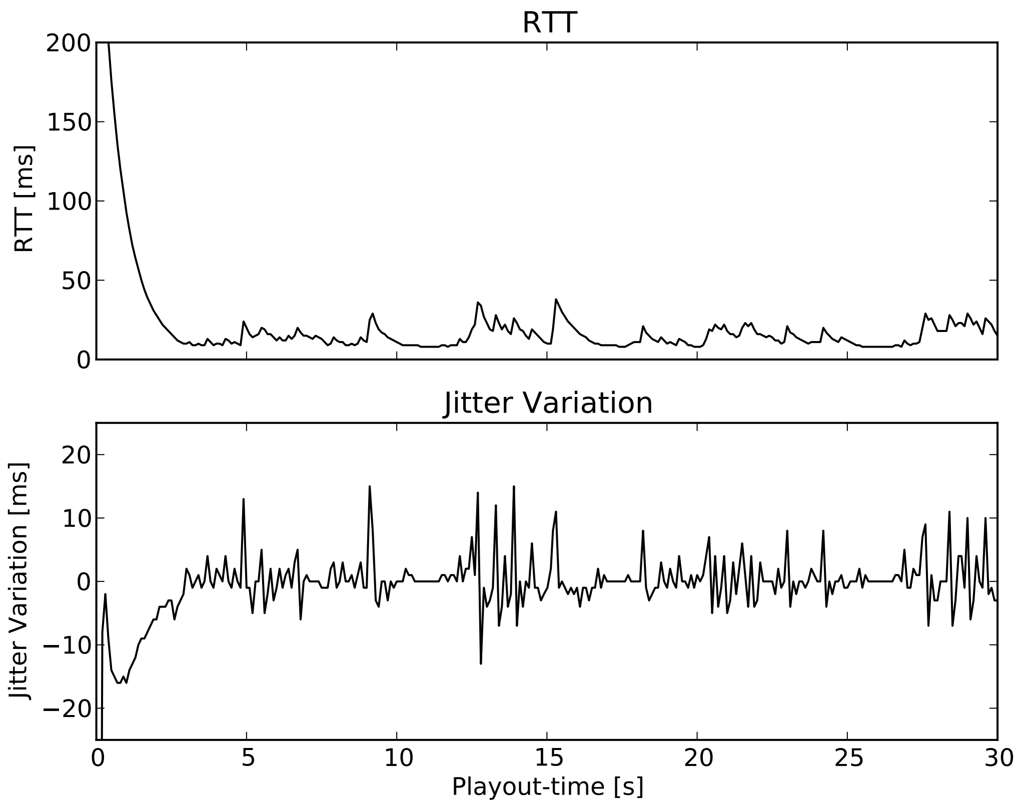


Figure 9: *RTT of a scaled video stream on a link with 700 kbps available bandwidth*

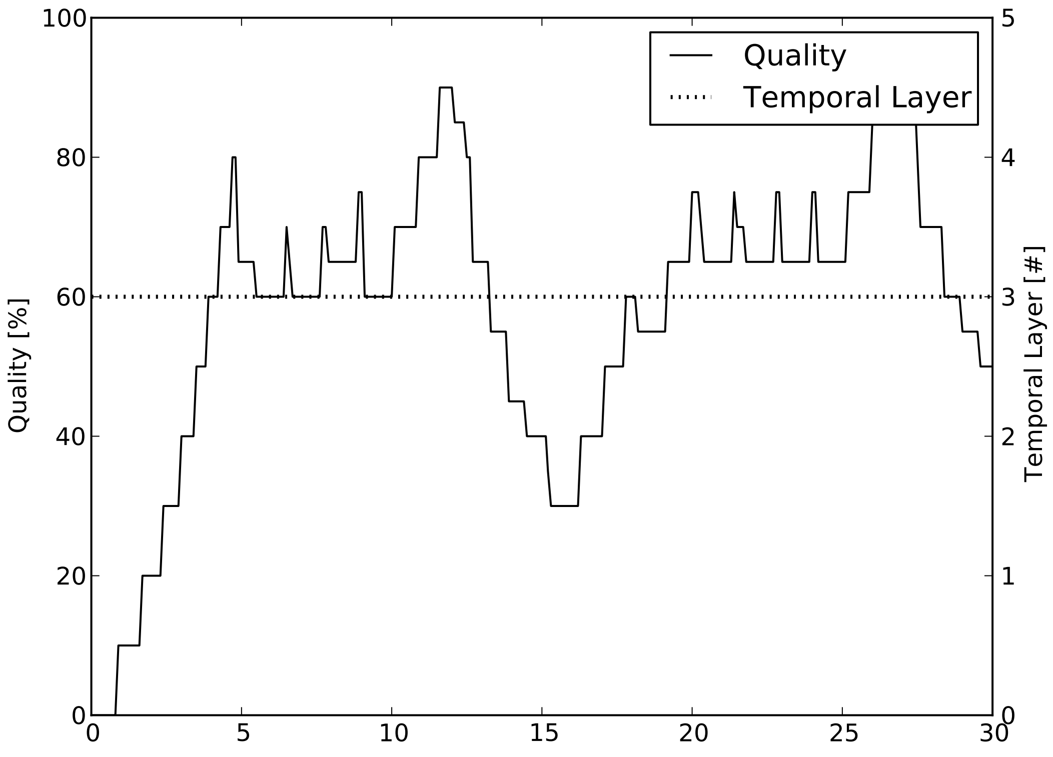


Figure 10: *Video stream adaptation on the sender-side*

bandwidth. The RTT is much lower than in Figure 8 and also the jitter variation fluctuates less.

Figure 10 shows the quality and temporal layer changes the sender applies to scale the video stream. In this scenario, it is not necessary to change the amount of temporal layer. At the 15s mark, where the complexity of the video stream rises, the sender lowers the quality and is able to avoid the congestion.

6.2 Receiver-Sided Video Scaling

The receiver uses the gap variation between the frames as indicator for changing network conditions.

Figure 11 shows the average gap between the frames at the receiver-side and at the sender-side. The gap at the sender-side is constant at 100ms (10 fps) most of the time, because it is

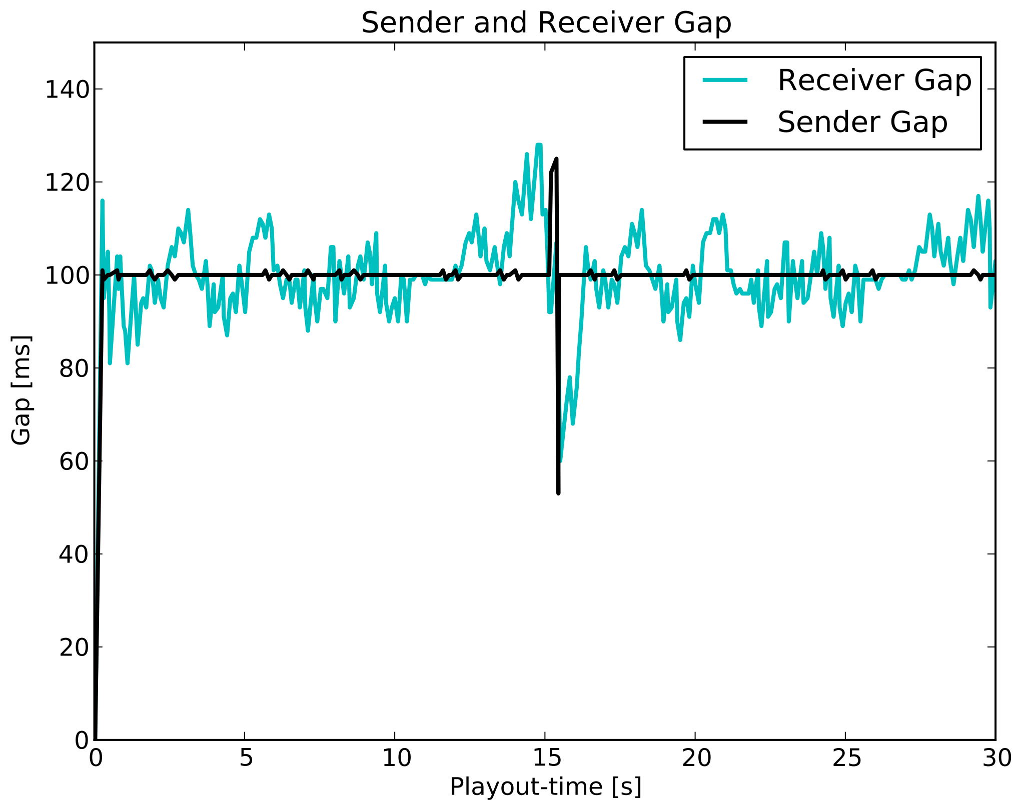


Figure 11: *Gap between the frames at the sender and at the receiver-side*

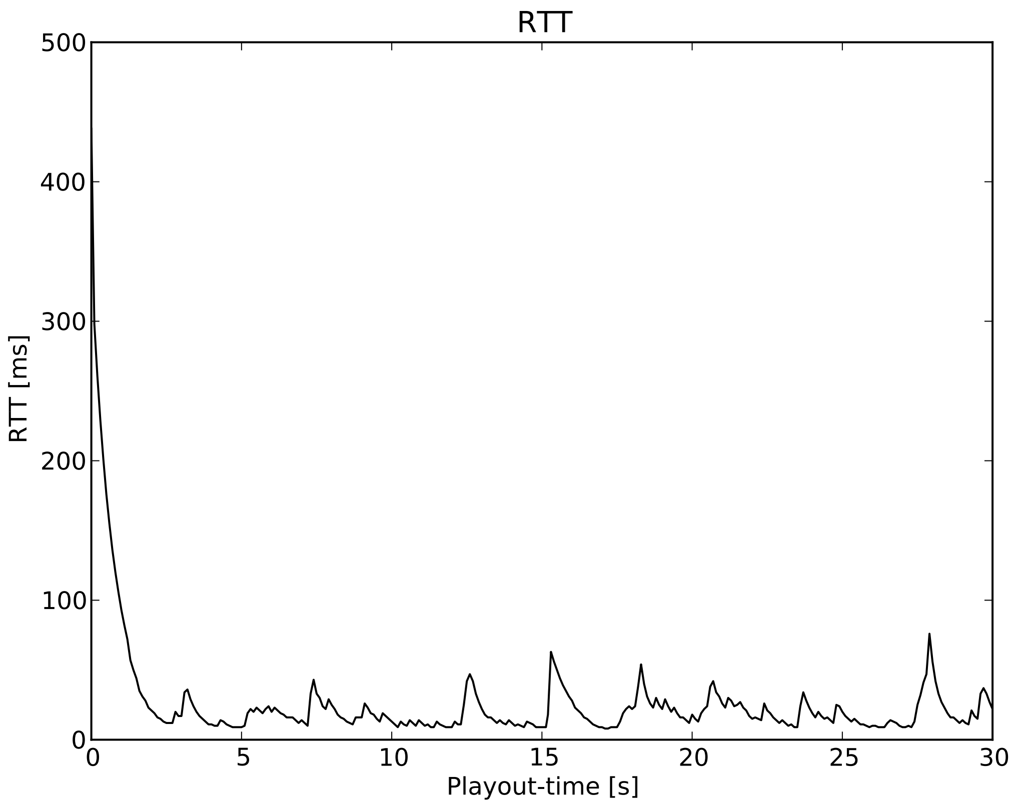


Figure 12: *RTT of a scaled video stream on a link with 700 kbps available bandwidth*

not influenced by the network conditions. The only variation is at the 15 second mark, where a new intra frame is encoded. At this point, the processor is not fast enough to encoded the frame in time and the frame is sent with a delay. The gap at the receiver-side has much more fluctuation (between 80ms and 120ms), because it is influenced by the network conditions.

Figure 12 shows the RTT of the receiver scaled video stream and Figure13 shows the gap between the frames on the sender and on the receiver-side for the scaled video stream. The RTT in the scaled stream is much lower than in the unscaled video stream, which indicates the receiver is able to scale the video stream. The average gap for the scaled video stream does not differ significantly. This is the expected behavior, because the receiver uses the increasing gaps as indicator for a congestion and therefore reacts after they increased.

Figure 14 shows the video codec scaling. In this scenario it is sufficient to scale only with the quality layer to scale the video stream. At the 15s mark, where intra-frame is encoded, the receiver recognizes the raising bandwidth requirements and suggests a downscaling to the sender. Compared to the sender-sided approach in Figure 10, the receiver-sided approach

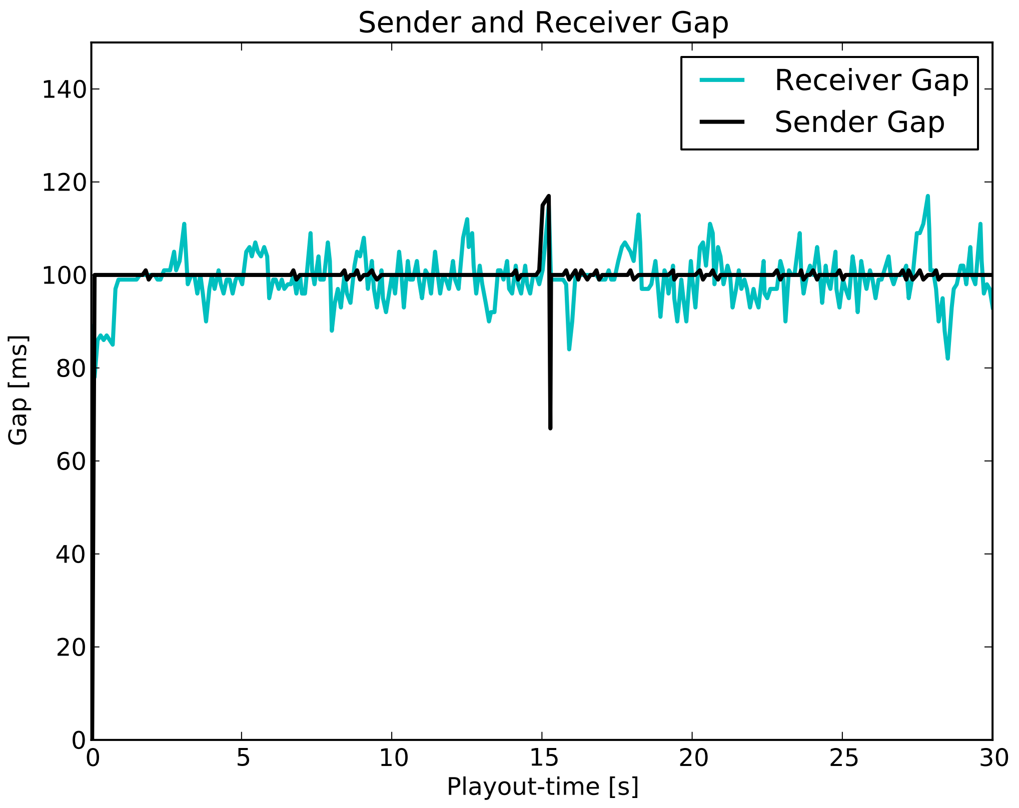


Figure 13: *Gap between the frames on the sender and on the receiver-side*

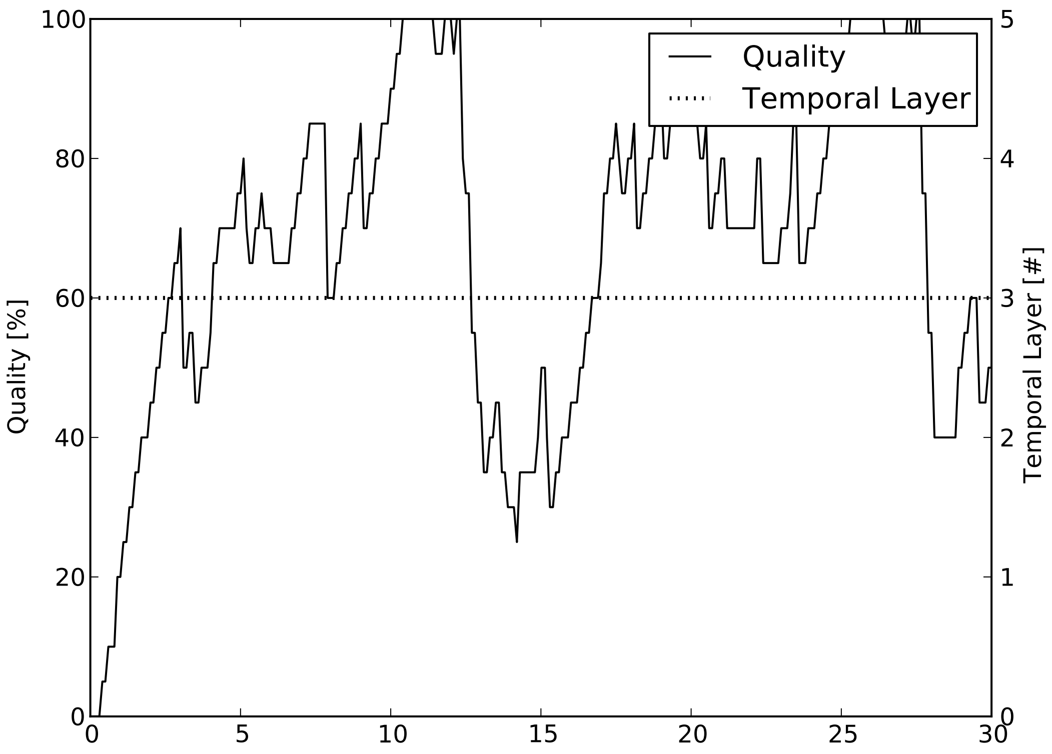


Figure 14: *Video stream adaptation on the receiver-side*

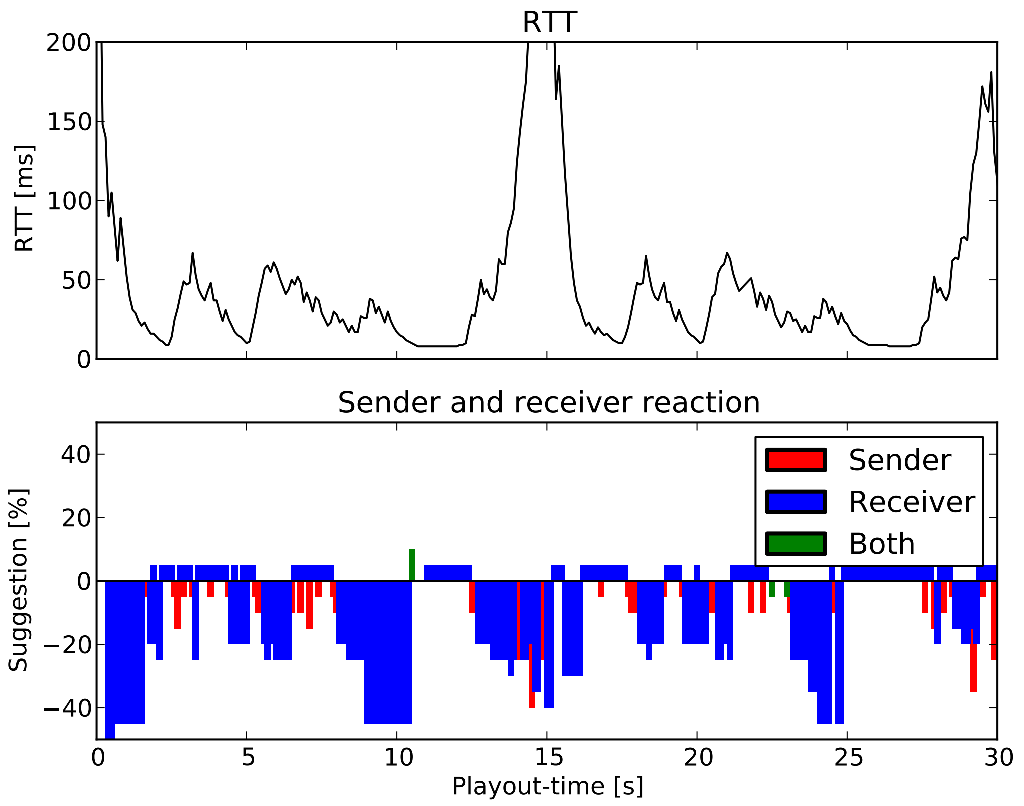


Figure 15: *Quality change suggestions of the sender and the receiver for an unscaled video stream*

has a more steeply increased video quality. The quality is not decreased as often as the sender does it and if the receiver decreases the quality, the changes are more radical.

6.3 Comparison between Sender and Receiver-Sided Video Codec Scaling

For a comparison of the two approaches, the scaling is disabled in this scenario and the quality is set fix to a high quality. Therefore, the quality change suggestions from the sender and receiver have no effect. This is necessary to analyze how both approach would react on the same network conditions.

Figure 15 shows the quality change suggestions from the receiver and the sender. Also the RTT of the video stream is shown for a rough visualization of the conditions on the path.

The blue bars are the suggestions of the receiver and the red bars are the suggestions of the sender. The graphs show, that a quality increase is only proposed by the receiver. The receiver is able to detect a free link and can safely increase the quality as described in Section 4.2.

The quality decreasing to avoid a congestion is more often proposed by the receiver, but the sender is always a bit faster than the receiver. However, in this scenario the suggestions are dropped and the video stream is not scaled. Therefore, the link still congests when the sender suggests a quality reduction. This congestions is detected by the receiver and it suggests a quality reduction as long as the link is congested. In this period the sender scaling does not work anymore, because it can not scale reliable on a congested link, as described in Section 3.3.

At certain points in time, the sender proposes a quality reduction while the receiver would propose a quality increase. In this situations the sender detects changes in the RTT and suggests a downscale, while the receiver does not see this small changes in the gap variation and suggests upscaling.

6.4 Video Codec Adaptation with Sender and Receiver

The aim of this work is a scaling software with both scaling approaches cooperating. In a real network, like the Internet, links with no delay are rather unusual. Therefore the software analyzes the video stream on a link with no delay, with 100ms delay and with 200ms delay.

6.4.1 Sender and Receiver scaling on a link with no delay

In this scenario both approaches are used to scale the video stream on a link with no delay. Figure 16 shows the RTT and the jitter variation of the scaled video stream. The RTT stays below 200ms and no congestion occurs. Figure 17 shows the gap between the frames on the sender and on the receiver-side for the scaled video stream.

In this scenario both, the sender and the receiver propose quality changes and the sender has to decide how to scale the video stream like described in Section 4.4.1. The resulting quality adaptation in Figure 18 is the result of both scaling proposes to the sender.

Figure 19 shows, who of the sender and receiver has caused the quality adaptation at a certain point in time. Most of the time the sender is responsible for the downscaling. But sometimes the quality reduction of the sender is not sufficient enough and in this cases the receiver recognizes the increasing frame transmission time and also downscales the video stream.

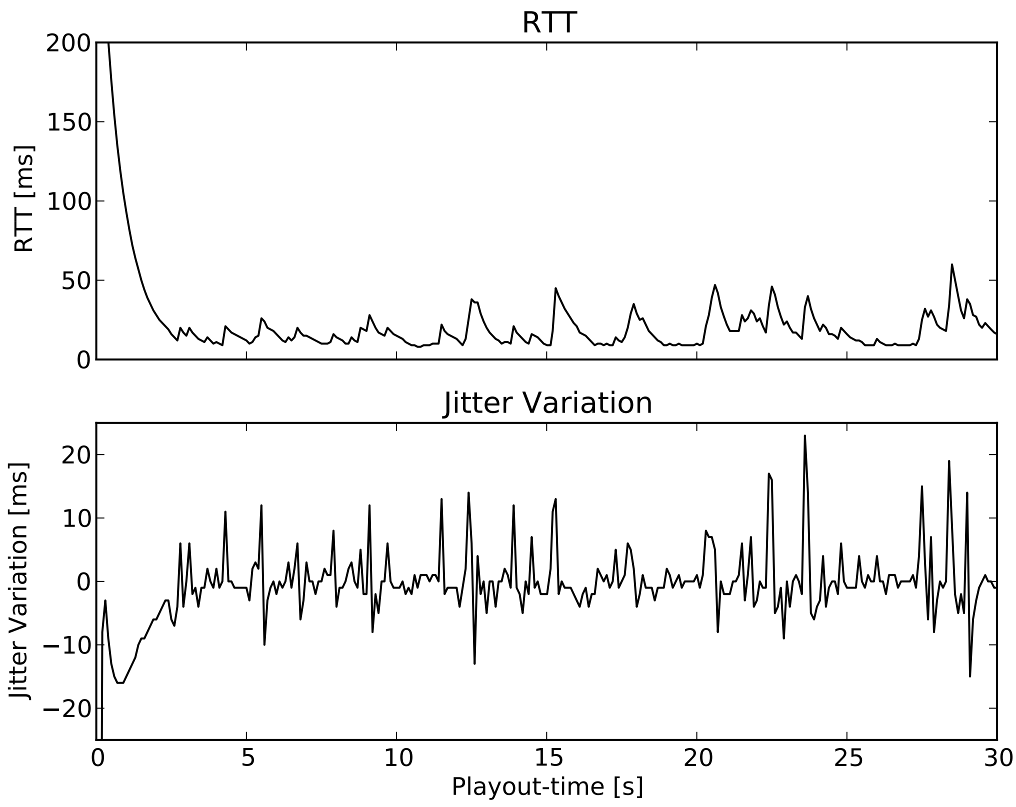


Figure 16: *RTT of an scaled video stream on a link with 700 kbps available bandwidth*

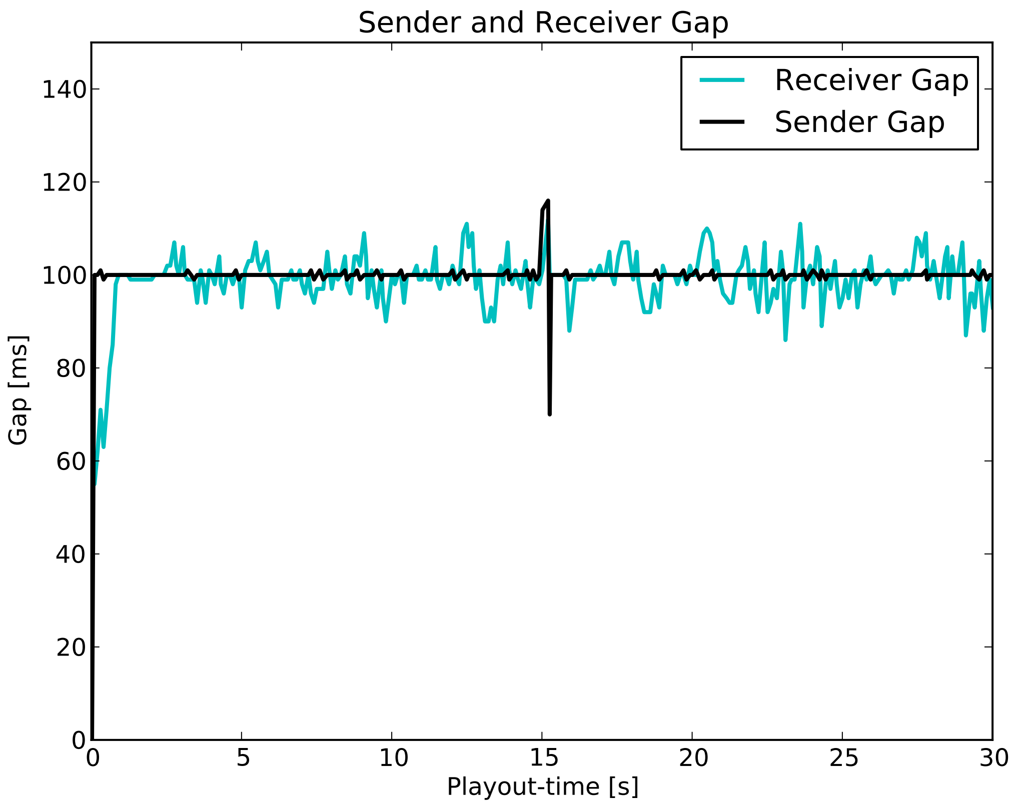


Figure 17: *Gap between the frames on the sender and on the receiver-side*

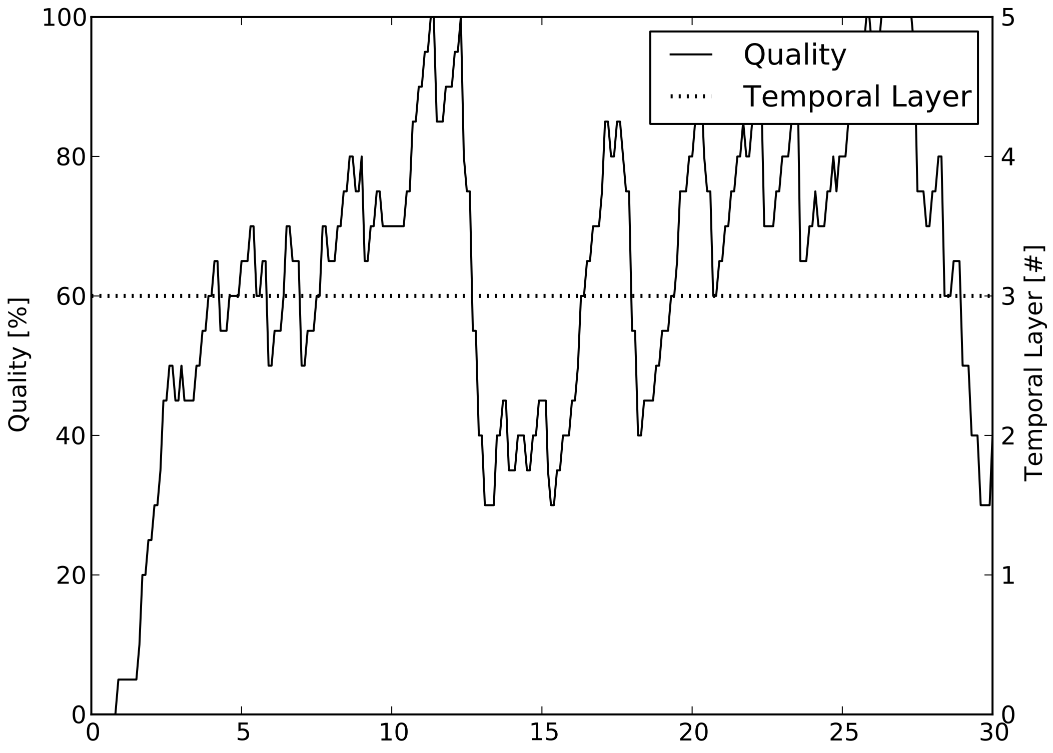


Figure 18: *Video stream adaptation on the sender and on the receiver-side*

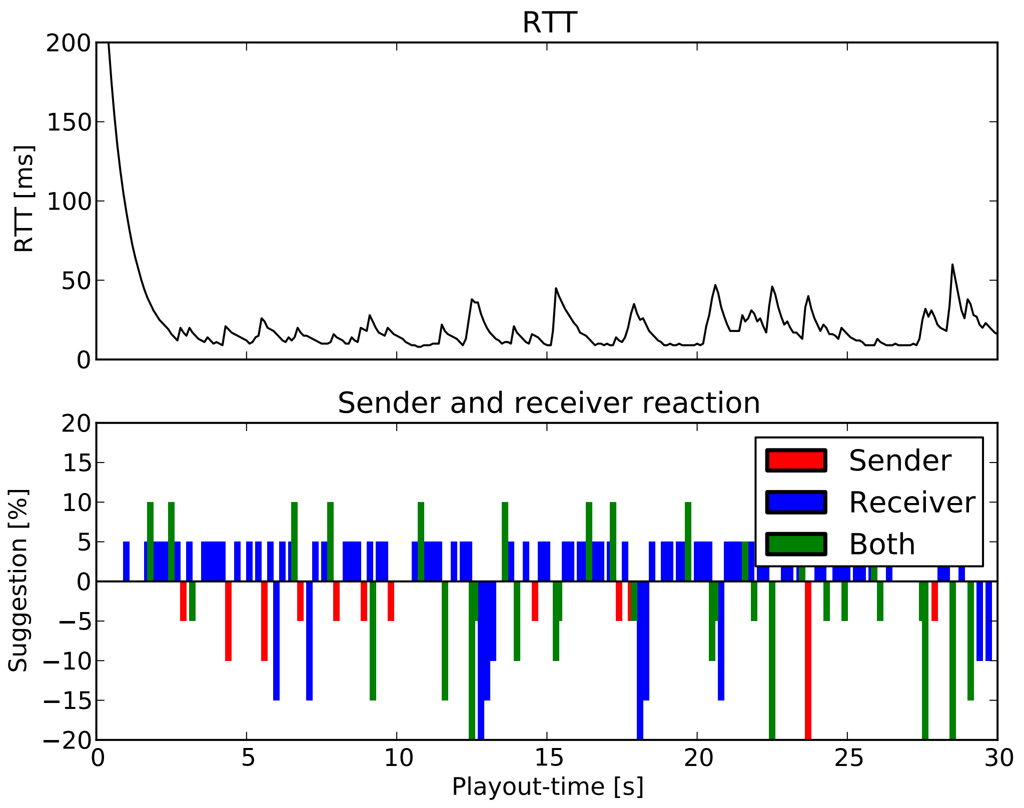


Figure 19: *Sender and receiver reaction*

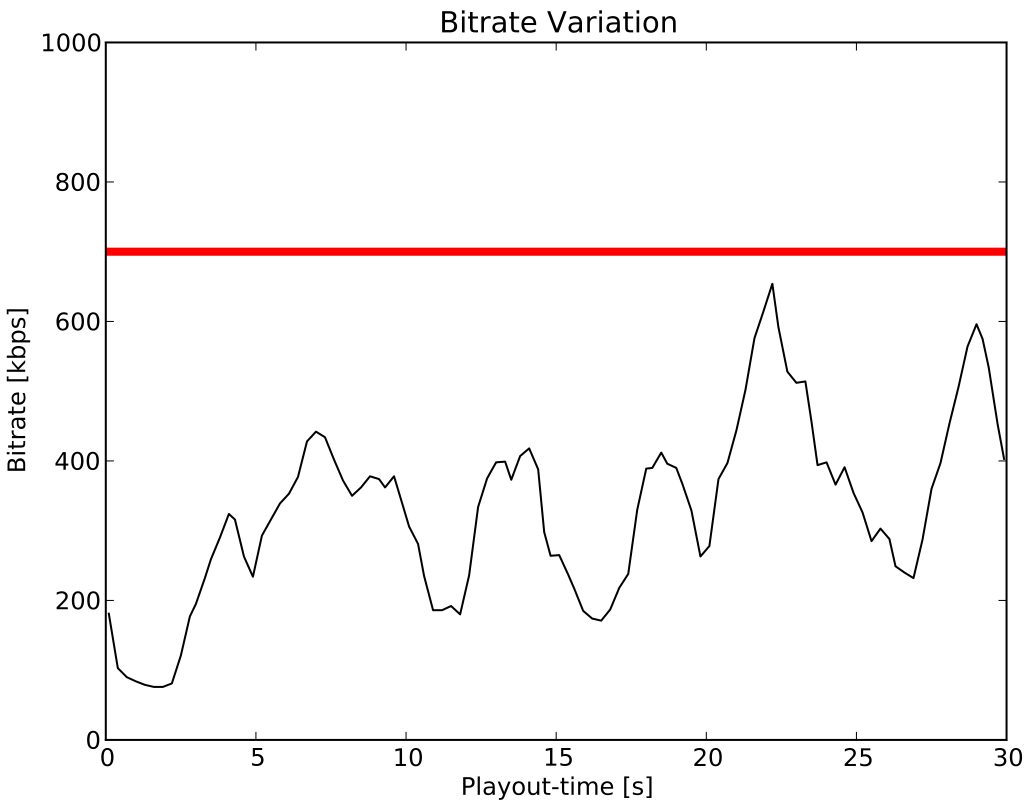


Figure 20: *Bitrate*

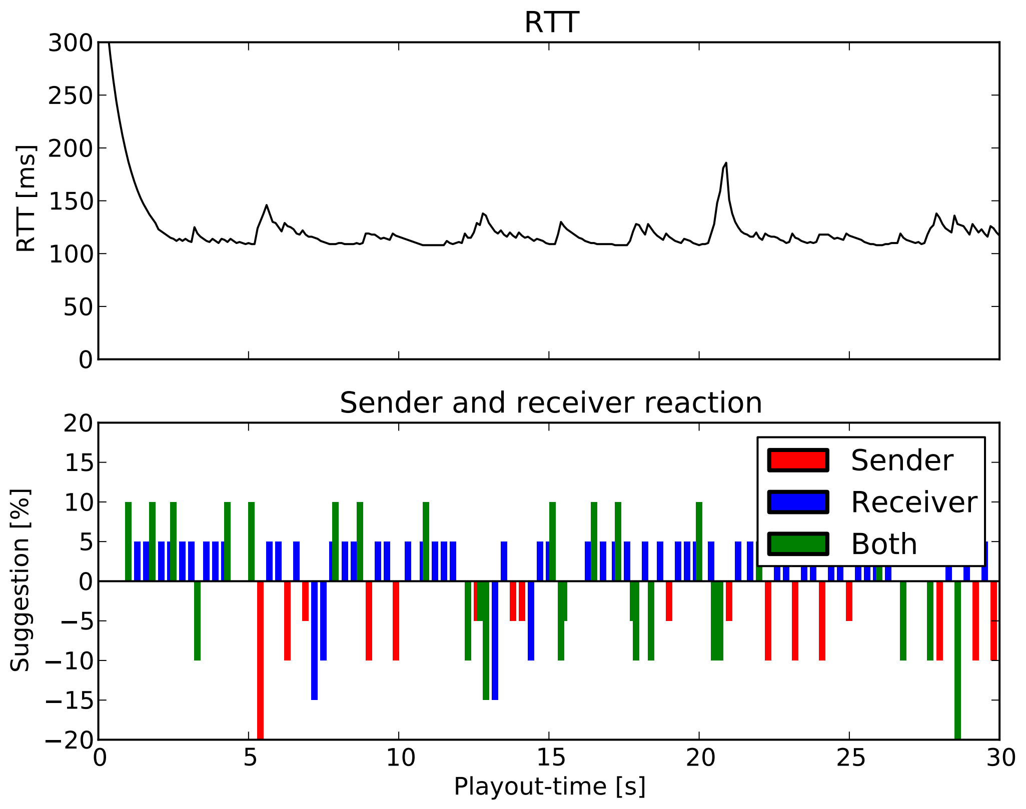


Figure 21: *Quality change suggestions of the sender and the receiver on a link with 100ms delay*

Figure 20 shows the average bitrate for the scaled video stream. It never exceeds the 700 kbps available bandwidth for the whole time and therefore does not congest the link.

6.4.2 Sender and Receiver scaling on a link with 100ms delay

For a link with 100ms delay, the reactions of the sender and receiver are shown in Figure 21. The behavior does not significantly differ from the link with no delay. The sender still reacts faster and more often with quality reductions than the receiver, while the receiver is responsible for the quality increases.

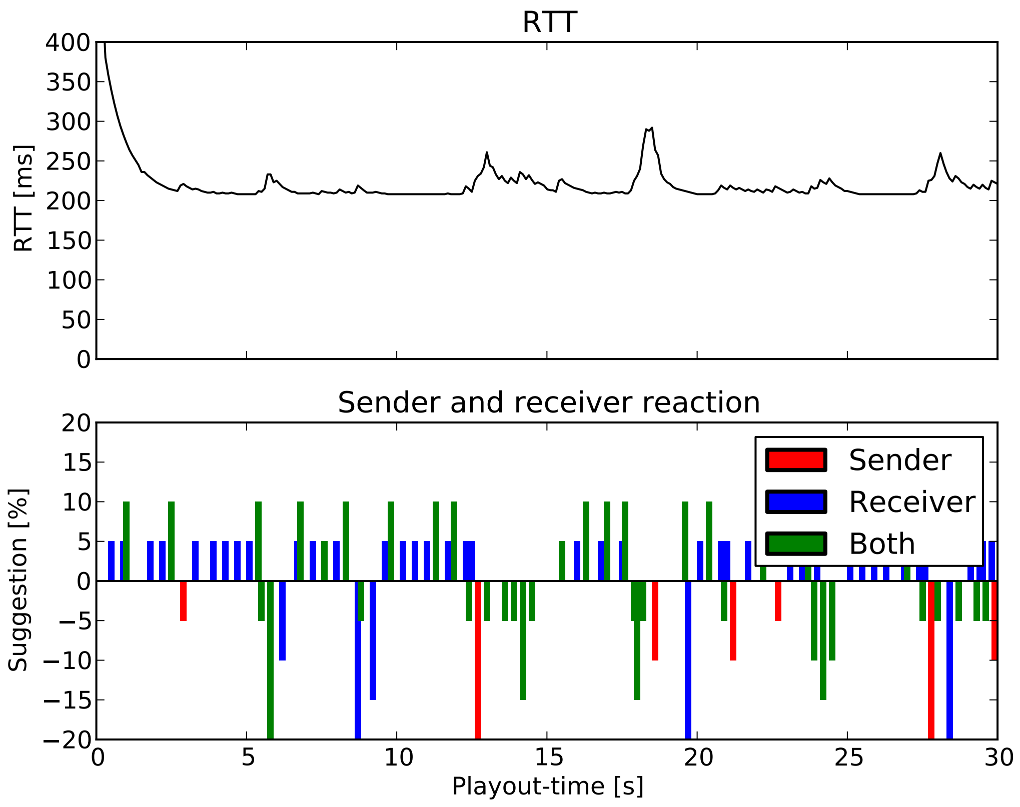


Figure 22: *Quality change suggestions of the sender and the receiver on a link with 200ms delay*

6.4.3 Sender and Receiver scaling on a link with 200ms delay

On a link with 200ms delay, the behavior of the scaling changes, which is shown in Figure 22. In this scenario, the receiver is still responsible for the quality increases, but the quality decreasing behavior differs from the former scenarios in Section 6.4.1 and 6.4.2. The sender does not react as much as before and the receiver does more downscaling. In this scenario it takes at least 400ms (2*one-way delay) for the sender to obtain information about the network conditions after sending a frame. In the 400ms without information about the network conditions, 4 frames are sent (10 fps) and might congest the path. The receiver does not rely on fast reactions for a reliable scaling and therefore is able to scale the video stream even with 200ms one-way delay. The reaction time is very important for the sender scaling, while the receiver is much more reliable in this scenarios.

7 Conclusion and Outlook

Multimedia applications with high quality video streams need to be aware of the network conditions to ensure a good utilized network path. In this work, we have presented a combination of a receiver-sided frame based approach, and a sender-sided packet based approach. Since neither of them measures the exact bandwidth, we also had to find a new approach to adapt the video codec.

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 and on the application layer. We analyzed the results from the sender and the receiver and used a combination of both to scale the video codec for an optimal bandwidth usage. Our test results showed, that this approach ensures a fast, reliable and accurate link observation and is capable of recognizing a link congestion early enough to avoid it before it gets noticeable for 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, while the receiver adaptation adds additional reliability in cases where the sender-sided approach might fail. In the future, we will also consider the spatial and quality layer for scaling and also factor the QoE into the scaling decision. We will examine the effects of packet-dropping and also use PlanetLab [17] to test our approach in a real network with multiple nodes and different network conditions.

References

[1] Saamer Akhshabi, Ali C. Begen, and Constantine Dovrolis. An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http. In *Proceedings of the second annual ACM conference on Multimedia systems*, MMSys '11, pages 157–168, New York, NY, USA, 2011. ACM.

[2] Hans L. Cycon, Thomas C. Schmidt, Gabriel Hege, Matthias Wählisch, Detlev Marpe, and Mark Palkow. Peer-to-Peer Videoconferencing with H.264 Software Codec for Mobiles. In Ramesh Jain and Mohan Kumar, editors, *WoWMoM08 – The 9th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks – Workshop on Mobile Video Delivery (MoViD)*, pages 1–6, Piscataway, NJ, USA, June 2008. IEEE, IEEE Press.

[3] Hans L. Cycon, Thomas C. Schmidt, Matthias Wählisch, Detlev Marpe, and Martin 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*, 57(3):1408–1415, August 2011. First place in 2012 Annual IEEE Consumer Electronics Society Chester Sall Memorial Award.

[4] L. De Vito, S. Rapuano, and L. Tomaciello. One-way delay measurement: State of the art. *Instrumentation and Measurement, IEEE Transactions on*, 57(12):2742–2750, December 2008.

[5] Enet. 25.03.2013. http://enet.bespin.org.

[6] Open Source Media Framework. 25.03.2013. http://www.osmf.org.

[7] Emanuele Goldoni and Marco Schivi. End-to-end available bandwidth estimation tools, an experimental comparison. In *Proceedings of the Second international conference on Traffic Monitoring and Analysis*, TMA'10, pages 171–182, Berlin, Heidelberg, 2010. Springer-Verlag.

[8] Cesar D. Guerrero and Miguel A. Labrador. On the applicability of available bandwidth estimation techniques and tools. *Comput. Commun.*, 33(1):11–22, 2010.

[9] Ningning Hu and Peter Steenkiste. Evaluation and characterization of available bandwidth probing techniques. *IEEE Journal on Selected Areas in Communications*, 21:879–894, 2003.

[10] ITU. G.114 - One-way transmission time. Recommendation - telecommunication union standardization sector, ITU, 05 2003.

[11] ITU-T Rec. H.264 and ISO/IEC 14496-10 (MPEG-4 AVC). Advanced Video Coding for Generic Audiovisual Services. Technical Report 8, ITU, July 2007.

[12] V. Jacobson. Congestion Avoidance and Control. *SIGCOMM Comput. Commun. Rev.*, 18(4):314–329, August 1988.

[13] Van Jacobson, Bob Braden, and Dave Borman. TCP Extensions for High Performance. RFC 1323, IETF, May 1992.

[14] Fabian Jäger, Thomas C. Schmidt, and Matthias Wählisch. Predictive Video Scaling - Adapting Source Coding to Early Network Congestion Indicators. In *2nd IEEE International Conference on Consumer Electronics - Berlin (ICCE-Berlin 2012)*, Piscataway, NJ, USA, Sep. 2012. IEEE Press.

[15] Dieu Thanh Nguyen and J. Ostermann. Congestion control for scalable video streaming using the scalability extension of h.264/avc. *Selected Topics in Signal Processing, IEEE Journal of*, 1(2):246 –253, aug. 2007.

[16] Roger Pantos. HTTP Live Streaming. Internet-Draft – work in progress 11, IETF, April 2013.

[17] PlanetLab. 25.03.2013. http://www.planet-lab.org.

[18] Open Video Player. 25.03.2013. http://openvideoplayer.sourceforge.net.

[19] Thomas Schierl, Thomas Stockhammer, and Thomas Wiegand. Mobile Video Transmission Using Scalable Video Coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1204–1217, September 2007.

[20] Henning Schulzrinne, Stephen L. Casner, Ron Frederick, and Van Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 1889, IETF, January 1996.

[21] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120, September 2007.

[22] Chan-Won Seo, Jung-Won Kang, Jong-Ki Han, and T.Q. Nguyen. Efficient bit allocation and rate control algorithms for hierarchical video coding. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(9):1210–1223, 2010.

[23] G.J. Sullivan, J. Ohm, Woo-Jin Han, and T. Wiegand. Overview of the high efficiency video coding (hevc) standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 22(12):1649 –1668, December 2012.

[24] Guillaume Urvoy-Keller, Taoufik En-Najjary, and Alessandro Sorniotti. Operational comparison of available bandwidth estimation tools. *SIGCOMM Comput. Commun. Rev.*, 38(1):39–42, January 2008.

[25] Wondershaper. 25.03.2013. http://lartc.org/wondershaper.