Masterarbeit

Fabian Jäger

How to Keep my Video Chat Fluent: An Approach to Adapting Scalable Video Flows to Heterogeneous Network Conditions in Real-time

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Fabian Jäger

How to Keep my Video Chat Fluent: An Approach to Adapting Scalable Video Flows to Heterogeneous Network Conditions in Real-time

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas C. Schmidt
Zweitgutachter: Prof. Dr. Franz Korf

Abgegeben am 26.03.2014

Fabian Jäger

Titel der Masterarbeit

Wie realisiert man eine stabile Videokonferenz: Ein Ansatz Videoströme in heterogenen Netzwerken in Echtzeit zu skalieren

Stichworte

Echtzeit-Videokonferenz, H.264, SVC, Bandbreitenabschätzung, RTP, Videoadaption, Videokonferenz

Kurzzusammenfassung

Videokonferenzen (VCoIP) sind ein zunehmender Trend im Internet. Dies gilt besonders für mobile Netzwerke, in denen die Bedingungen sehr unbeständig sind und die Videokonferenz beeinträchtigen. Eine Möglichkeit zur Behebung des Problems ist ein Scalable Video Coding (SVC) Videostrom, der eine individuelle Adaption an die vorhandene Bandbreite erlaubt. Dafür muss die verfügbare Bandbreite zunächst abgeschätzt werden. In einer echtzeit Videokonferenz-Software muss dies schnell und zuverlässig geschehen. In dieser Arbeit werden die Möglichkeiten für den Sender und Empfänger untersucht, die Bandbreiten auf einem Pfad abzuschätzen. Auf der Senderseite wird dazu die Variation des RTT Jitters untersucht. Auf der Empfängerseite wird die Variation der Empfangszeitpunkte für Frames analysiert, um Rückschlüsse auf die Netzwerkbedingungen zu ziehen. Die Adaptionsverfahren werden in unterschiedlichen Netzwerkbedingungen untersucht und zeigen eine schnellere Reaktion auf der Senderseite, während der Empfänger zuverlässigere Ergebnisse erzielt.

Title of the master thesis

How to Keep my Video Chat Fluent: An Approach to Adapting Scalable Video Flows to Heterogeneous Network Conditions in Real-time

Keywords

Real-time videoconferencing, H.264, SVC, Bandwidth adaptation, RTP, Video adaptation, Videoconferencing

Abstract

Video conferencing over IP (VCoIP) is a major trend in current Internet communication and has particularly spread to the mobile realm. In this environment, users face the problem of heterogeneous and fluctuating network conditions. A promising solution to this issue is the scalable video coding (SVC). It allows an adaptation of the video stream to the available bandwidth, but requires a reliable bandwidth estimation. Adaptation times for conversational video at fluctuating network conditions are critical, and a fast strategy for bandwidth estimation is needed to avoid congestion. In this work, we analyze the capabilities of the sender and the receiver to adapt the video coding to changing network conditions. We derive an early congestion indicator at the sender side based on the jitter variation. For receivers, we use the inter-arrival jitter to extract a feasible scaling. The video adaptation approaches are evaluated in different network conditions and reveal a faster congestion detection at the sender, while scaling rules work more reliably at the receiver.

Contents

List of Figures

List of Tables

1 Introduction

Video communication is one of the fastest growing phenomena on the Internet. It has been implemented in many applications like conferencing software, online games, instant massaging, and mobile applications. Conversational video inherits delay sensitivity from audio, while its large bandwidth consumption may easily cause or amplify congestion on the communication path. A congested path can lead to transmission errors like packet loss, delay and jitter that typically degrade the visual quality or stall the video flow. Heterogeneous and varying network conditions, which are common in mobile environments, increase the likelihood of congestions.

Keeping the video fluent and at appropriate quality requires a dynamic adaptation, whenever network services fluctuate. The current video coding standards H.264/AVC [1] and H.265 [54] allow for a dynamic rescaling in time (adaptive framerate) and quality (adaptive quantization). The scalable video coding extensions [52] enable additional spatial scaling and arrange layers in packet streams. All codecs need a proper parametrization in order to establish a robust and reliable video conversation.

Appropriate video scaling is derived from estimating the effective bandwidth currently available in the network. Such estimators are required to detect congestion as early as possible and to predict a bitrate that complies to the network constraints of the immediate future. Network congestions arise at overloaded network elements when the overall transmission demands exceed the available bandwidth along the path. It is neither easy to identify the available bandwidth, nor to determine a congested link in real-time. In general, this can be done at the sender or at the receiver side, which have access to different measures of network performance.

The Internet is a highly distributed network that does not guarantee end-to-end performance. Packet delivery delays are often caused by a congested path between sender and receiver. However, avoiding a stuttering video stream is important to provide a good user experience during a video conference. To leverage modern video codecs, which can scale to meet available network resources, three basic tasks need to be supported: (a) detect a congested path, (b) approximate the available bandwidth, and (c) adjust the codec accordingly. The time to identify congestion and estimate the bandwidth is crucial. When a congestion is detected, the ratio of available bandwidth to the bitrate of the video stream needs to be estimated to scale down the video codec by the same factor.

Any solution that gives input to parametrize the video codec should not introduce additional complexity to the network. Dedicated probing techniques (e.g., to explore the bandwidth) should be

avoided as they increase congestion in large-scale deployments. The change of network protocols should not be part of the solution space as this prevents easy deployment. The challenges in designing a solution are twofold. First, a congested link needs to be detected as fast as possible on the basis of available network metrics. Second, an approximation of proper codec scaling needs to be derived.

In this thesis, we explore methods to detect changing network conditions at the sender and the receiver side, as well as corresponding strategies for an appropriate video scaling. In detail, our contributions read:

1. An algorithm for early congestion indication at the sender based on the jitter variation observed from a fast feedback loop of the transport protocol.
2. An algorithm for estimating current network conditions at the receiver based on the inter-arrival jitter. This approach requires an extra feedback channel to rescale the sender.
3. Adaptation strategies for video scaling based on the estimates of the sender, the receiver, and a combination of both.

The remainder of this thesis is organized as follows. Section 2 discusses the problems and requirements for an adaptive video conferencing software and gives references to related work. Our core algorithms for detecting congestion and estimating appropriate scaling parameters is presented in Section 3. These approaches are applied and extensively evaluated with the help of our full-fledged video conferencing software system as documented in Section 4. Section 5 analysis the adpatation approaches and discusses the experimental avaluations. We conclude with an outlook in Section 6.

2 Adaptive Video Scaling and Related Work

The objective of this work overlaps with several independent research areas, most notably (1) scalable video deployment in heterogeneous regimes, (2) bandwidth estimation techniques, (3) approaches to adaptive scaling at receiver and sender, and (4) flow control in the presence of competing traffic. Each of them influence the video stream transmission and needs to be researched to develop video adaptation strategies. Thus, the following subjects are examined:

- Video codecs, video stream scalability, and video rating metrics
- Common protocols that are used in multimedia applications
- Adapting a video stream to the network conditions

2.1 Video Codecs and Video Rating Metrics

The available bandwidth of networks is often a limiting factor in multimedia streaming scenarios. Especially real-time multimedia applications suffer. The usage of a video codec to reduce the bandwidth demands is essential to provide a high quality video stream in a network with low available bandwidth. The advantage of encoded video streams over raw video data is the decreased bandwidth requirements, which mainly depend on the codec in use, the encoding quality, and the compression complexity.

2.1.1 Video Codec and their Scalability

In general, a video codec compresses video data (usually lossy). Every frame of an encoded video is encoded with common image compression algorithms (eg., JPEG [35]), which reduces extraneous or duplicate information in the frames. In comparison with image compressing algorithms, video codecs are also able to detect and reduce redundant information in the temporal scale. Only the parts of a frame that change over time are transmitted.

Most common video codecs have a similar process flow to encode a video stream. First, a complete frame is encoded that has no dependencies to other frames, named *Intra-frame*. All following frames, named *Inter-frames*, depend on it. This goes on, until a new *Intra-frame* is

encoded. An *Intra-frame* and all depending frames is shown in Figure 2.1 and is named a *Group of Pictures* (GOP). In this example a simple structure is shown, where every *Inter-frames* depend only on its predecessor, but it is also possible that *Inter-frames* depend on multiple frames in the same GOP. Multiple dependencies reduces the produced bit rate, but also increases the encoding

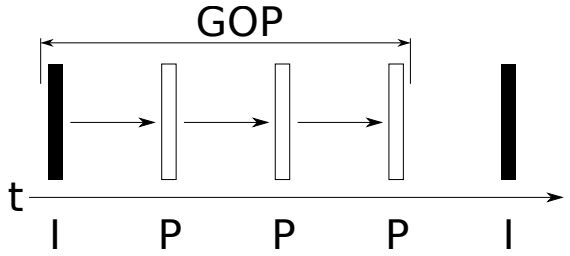


Figure 2.1: *Group of Pictures with one Intra-frame (I) and 3 Inter-frames (P)*

complexity and more processing power is required. In real-time multimedia applications, the frames usually depend only on the previous frames. If they also depend on following frames, they need to be buffered, which delays the playout. For a correct playout at the receiver-side, all frames are required. If a frame is missing, the depending frames cannot be correctly decoded.

For a fluent video stream, it is necessary to scale the video stream to prevent bandwidth exhaustion. This holds in particular for mobile regimes [50], [10] and can be achieved with the scalable video coding (SVC) [52], which is an extension to H.264 [1]. In this work, we focus on the H.264 and its SVC extension, which allows to scale the datarate and coding complexity of the video for each participant individually [9]. It is worth mentioning that on the 25 January 2013 the new codec standard High Efficiency Video Coding (HEVC, ITU-T H.265 or ISO/IEC 23008-2 [54]) was released, which only needs half of the bit rate of its predecessor ITU-T H.264 [54]. The HEVC can provide a smaller video stream with the same quality as H.264, but also requires more processing power.

T. Schierl *et al.* [50] present an overview of basic solutions for the deployment of scalable video streams in mobile realms, regarding various set-ups for IP and non-IP worlds. Particular focus is given by the authors to the problems of unstable network conditions with significant packet loss. As such, conversational or broadcast-type multimedia applications suffer from varying throughput and a scalable video stream bears potentials to adapt to the poor network conditions. The authors present solutions of the SVC coding standard to face these problems and propose an approach for integrating the SVC in existing and emerging mobile networks. Insights are given on how the SVC interacts with mobile networks, different QoS metrics, and content delivery protocols. Furthermore, some use-cases are introduced and show the relation between the SVC and mobile delivery methods.

2.1.2 Rating Metrics for Video Streams

The network conditions influence the video stream and are usually percept as quality impairments. To measure the effects of network impairments, a rating metric is needed for the video streams.

The quality of the video transmission depends on the performance of the network, which is refered as Quality of service (QoS). Several aspects of the network service can be considered to rate the QoS, such as error rates, bandwidth, throughput, transmission delay, availability, jitter, etc. The objective video stream quality depends on the QoS and several objective metrics exist to rate the video quality. The most popular objective quality metrics are the peak signal-to-noise ratio (PSNR) [49], the mean squared error (MSE), the moving picture quality metric (MPQM) [57] and the normalized video fidelity metric (NVFM) [61]. However, the user's impression on video conferencing applications depends on the subjective perception rather than the objective QoS. Thus, the video stream is characterized in terms of Quality of Experience (QoE) rather than QoS [16].

A common way of subjectively characterizing a video is the *Mean Opinion Score* (MOS). It is standardized from the ITU in ITU-T Rec. P.910 [30]. The MOS is a measurement of the human perception and is measured when the video is played out and the user gives direct feedback to the perception of the quality. It is usually a scale between 1 and 5 and is a representation of the average human response to the quality of a video. Table 2.1 shows the interpretation for the MOS

MOS	Quality	Impairment
5	Excellent	Imperceptible
4	Good	Perceptible but not annoying
3	Fair	Slightly annoying
2	Poor	Annoying
1	Bad	Very annoying

Table 2.1: *MOS rating*

values.

The MOS metric has also drawbacks, which are notable [6]:

- Subjective quality is not a standardized metric and depends on the user's sensation.
- Users avoid extreme scores and most of the videos are rated as 2,3, or 4
- The users tend to 'forgive' short quality impairments, when the playout is long and smooth altogether

Video rating metrics (subjective and objective) can be classified in the following schemes:

Full reference (FR): Both, the outcome video and the original video are available and allow a detailed comparison. The video and its single frames can be compared with objective and subjective metrics. Full reference metrics require access to the original video and the received video, but also provide reliable analysis. A good example for a FR metric is PESQ [29] or the PSNR where the original picture is needed for the calculation.

No reference (NR): In NR metrics, only the outcome video is available and the original video itself is not accessible nor are any encoding or network parameter from the source. NR metrics lack the possibility to differentiate between quality impairment from the original video and additional disturbance by the network. NR metrics are a typical online streaming scenario, where a video stream is received without access to the original video. Typically, network measurements such as packet loss ration or jitter are used to evaluate the video quality.

Reduced reference (RR): In RR metrics, the original video is like in NR metrics not accessible, but certain parameters are extracted at the source and transmitted to the receiver. The parameters that are needed for the rating depend on the rating algorithm. The measured QoE metrics are less accurate than a FR metric but produce less overhead. RR metrics are often used in application, where the bandwidth utilization is crucial.

The most popular MOS metrics are the *Visiual Quality Metric VQM* [47], PEVQ [48], and the *Media Delivery Index* (MDI) [60]. VQM is a FR metric and provides an objective measurement of the perceived video quality. For this purpose, the perceptual effects of video impairments are measured. The measurements include blurring, jerky/unnatural motion, global noise, block distortion and color distortion, and combines them into a single metric. It is a fairly complex metric, but it also correlates very well with the human perception. VQM can also be optimized on the area of application like television or videoconferencing. PEVQ is a FR end-to-end measurement algorithm to calculate a MOS rating for a video sequence. It also provides various key performance indicators (KPIs) like PSNR, delay, jerkiness, and blur. PEVQ models the human visual system with the anomalies how they perceive the video signal. The algorithm basically consists of 4 components. The first component is responsible for the spatial and temporal alignment of the incoming stream. This ensures that only corresponding frames are compared to each other. The second component determines the perceptual difference in the video signal that is perceived by the user. The third component classifies the video signal and the forth component calculates a MOS rating based on the measurement of all components.

The MDI is a lightweight approach, which provides and indication of the expected QoE based on network level metrics. It consists of two components, the *Delay Factor* (DF) and the *Media Loss Rate* (MLR). We will present the DF component more detailed, because a similar approach is used to rate the videos in this work.

The MLR is the number of lost packets or packets that are out of order in one second. The MLR is calculated with equation (2.1),

$$MLR = \frac{P_e - P_r}{\delta} \tag{2.1}$$

where P_e is the number of expected packets, P_r is the number of received packets and δ is the time interval in seconds. The maximum acceptable loss rates are very demanding, because every packet loss causes significant visible artifacts in the decoded video. A maximum loss of up five consecutive IP packets per thirty minutes is recommended for SDTV and VOD and four hours for HDTV [38]. The resulting MLR values shown in Table 2.2.

Service	Max. Acceptable Average MLR
SDTV	0.004
VOD	0.004
HDTV	0.0005

Table 2.2: *Maximum acceptable loss rate for videos*

The DF estimates a QoE based on the network delay. The delays on the network have various reasons (eg., congestions or path changes), but in any case, if the arrival rate does not match the rate at which the destination is consuming data, the packets must be buffered. If packets arrive at such a high rate, that they fill the buffer completely, they cause an overflow and packets get dropped. If packets arrive too slow, the buffer underflows and not enough data is available to feed the decoder at the desired rate. Both situations are undesirable and lower the QoE. The DF component is a time value indicating how many miliseconds' worth of data the buffers must be able to contain to compensate the jitter in the arriving rate [38]. It is based on the buffer filling level at the receiver side and calculated with equation (2.2) and (2.3).

$$\Delta = |bytes\_received - bytes\_drained| \tag{2.2}$$

$$DF = \frac{(max(\Delta) - min(\Delta))}{media\_rate} \tag{2.3}$$

The minimum and maximum filling level of the buffer is measured and their difference is divided by the incoming media rate. A downside of the MDI is that it requires a buffer at the receiver-side to calculate the DF. In a video conferencing software, a buffered video stream is unwanted and therefore we will propose another approach to rate the delay from a QoE perspective.

T. Hendrawan and I. Mahadhika [24] examined the effects of packet loss to the QoE of a scalable video stream on a wireless network. They used a laptop, a tablet and a smartphone as receiver to analyze the QoE for the different devices. The coding software is the *Joint Scalable Video Model* (JSVM), which uses the H.264/SVC standard as reference. A video stream is encoded

with JSVM and sent to multiple receivers over a IEEE 802.11n network. The results show, that it is not negligible what kind of device is used to playout the video. The same video gets different MOS ratings on a smartphone than on a device with a bigger display like a laptop. Also, the compression ratio influences the QoE. A video stream with a high compression has less packet loss than a video stream with a low compression, but when a packet loss occur, the effects have more impact than a packet loss on a stream with a low compression.

2.2 Estimating Network Conditions and Available Bandwidth

Variable	Description
$\mu(t)$	Service rate of the router
$\lambda(t)$	Sending rate of the sender
c	Capacity of the path
$w(t)$	Packet dropping probability
$\eta(t)$	Competing traffic stream

Table 2.3: *Network notation*

The limiting factors for a video tranmsissions are the available bandwidth and the loss rate of the path. If the applications do not react to a changing available bandwidth or a changing loss rate, the path impend to congest. A detailed analysis of congestions and their course of events is important to develop strategies to prevent them. Table 2.3 summarizes the notation we use in the following sections.

2.2.1 Router Queues, Packet Loss, and Available Bandwidth

Congestions arise at routers if the incoming traffic exceeds the available bandwidth of the outgoing link and the router queues start to fill. The router queues are considered to be M/M/1/N queues, where N is the length of the queue [13] [36] . The router drop packets depending on the filling level of its queue, which inflicts the QoS of the traffic streams. For an unreliable traffic stream like UDP flows, the follow-up will be packet loss. The follow-up for a reliable traffic stream like TCP flows will be a retransmission of the lost packets, but the TCP protocol decreases the sending rate to avoid a further congesetion.

The decision, which packets are dropped is important to avoid congestions as best as possible. Active Queue Management (AQM) algorithms has been the focus of research for a long time. The

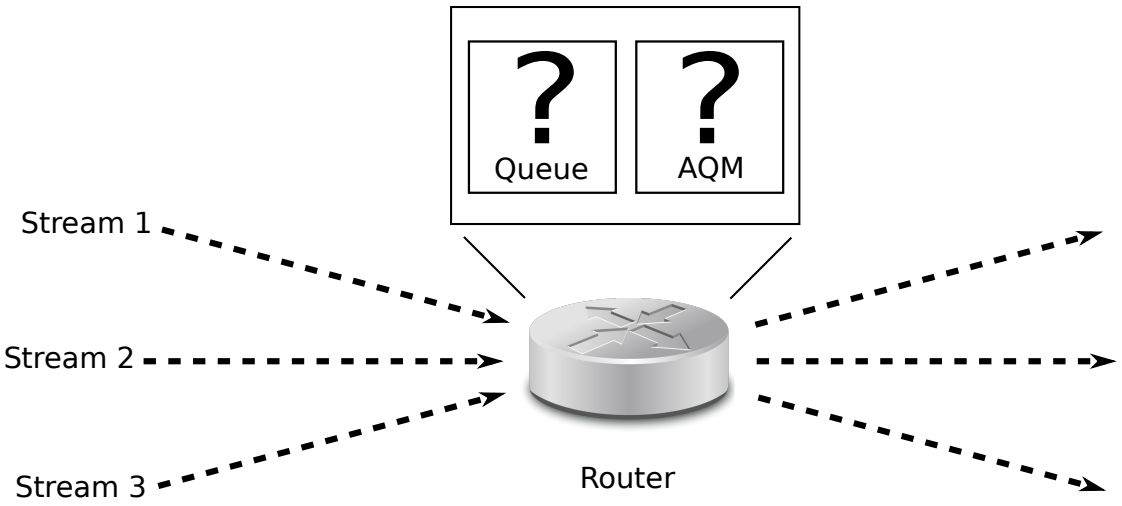


Figure 2.2: *The AQM and the queue status are not known for the traffic streams 1-3*

advantage of an AQM over a drop-tail (incoming packets are dropped if the queue is filled) approach is that it 'informs' the sender about an impending congestion by scattered packet dropping before buffers overflow. The senders are informed early on and can react accordingly. The AQM decides when, which, and how many packets are dropped. AQM algorithms can be classified in three categories: queue based, rated based and per-flow algorithms. Queue based algorithms calculate the dropping probability for incoming packets with the average queue length. Rate based algorithms do not take the queue length into account and calculate the dropping probability with the incoming and outgoing rates. Per-flow scheduling mechanisms calculate the dropping probability for each flow individually, which provides max-min fairness, but it is also necessary to keep the state for all flows.

The most common AQMs are RED[17], ARED[14], REM[3], BLUE[5] and modifications of these algorithms. The queue based algorithms (RED,ARED,REM) calculate the dropping rate with the average queue size. An empty queue means that enough bandwidth is available and the dropping probability for the incoming packets is low. With a growing queue, the dropping probability for the incoming packets increases. When the queue is full, the probability reaches its maximum and every incoming packet is dropped. The approaches mainly differ in the calculation of the dropping probability. BLUE has a rate base approach and calculates the dropping probability with the packet loss and link utilization rather than on queue lengths.

The AQM has significant effects on the queuing delay, but without access to the routers, the operating algorithm is unknown and therefore, the router is a blackbox (cf., Figure 2.2). In a real network, the bandwidth requirements from the competing traffics are unknown, but we can assume that the service rate is the capacity without the bandwidth consumed by the competing

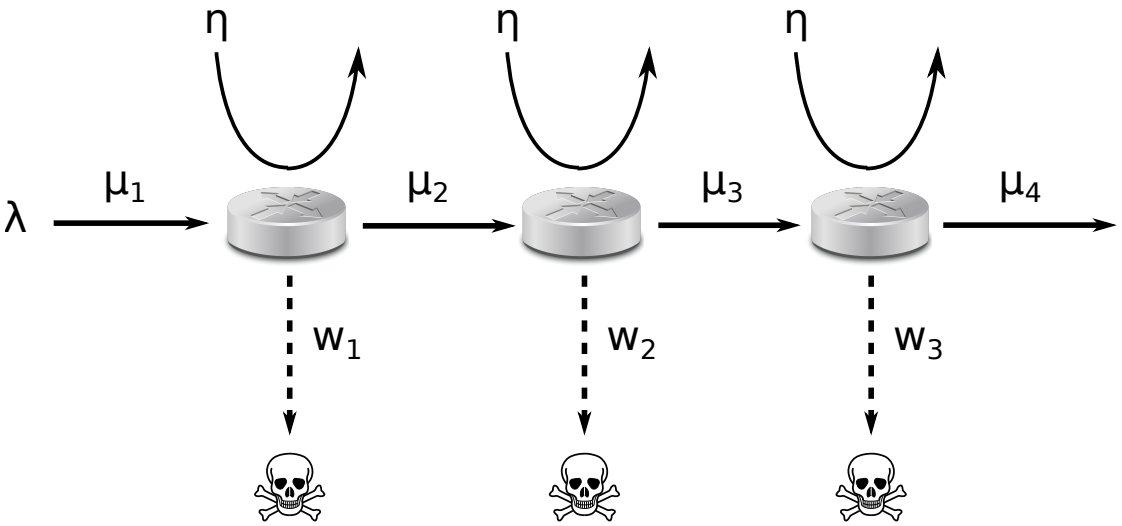


Figure 2.3: *Traffic stream on a path with multiple hops and competing traffic*

traffic streams. The service rate $\mu(t)$ for each traffic stream on each router is shown in equation (2.4),

$$\mu_i = c - \sum_{j=0}^{m(t)} \eta_j(t) \tag{2.4}$$

where $\eta(t)$ is the competing traffic and $m(t)$ is the amount of competing traffic streams. The minimum $m(t)$ is zero. If $\mu_i(t)$ is zero, the link is congested and the queuing delay increases.

Usually, a path consists of multiple routers, where the capacities and the competing traffic streams are unknown. This is shown in Figure 2.3. For the endpoints of a connection, the available bandwidth on each router are unknown without any further measurement. The available bandwidth on a path only depends on the tight link with the smallest available bandwidth. This is not necessarily located at the bottleneck (smallest capacity) of the path, but can occur on any link (cf., equation (2.5)).

$$\mu(t) = min(\{\mu_1(t), \mu_2(t), ..., \mu_n(t)\}) \tag{2.5}$$

If the available bandwidth $\mu(t)$ is zero, the link is congested. As a result, the queues fill up and increases the traversal time for the packets. If the situation does not change and the router queues impend to be completely filled, the AQM starts to drop packets.

In a multimedia application, the traffic streams usually contains video and audio data. The bit rate of a traffic stream does not change between the routers, although it may happen if IP fragmenta-

tion occurs. These small variations do not have significant influences and therefore, we assume that the bitrate $\lambda(t)$ does not change (cf., equation (2.6)).

$$\lambda(t) \approx \lambda_1(t) \approx \lambda_2(t) \approx ... \approx \lambda_n(t) \tag{2.6}$$

In this work, the traffic stream is a video stream with a few additional bytes of the transport protocol. This overhead is very small and the bit rate of the traffic stream $\lambda(t)$ is approximately the bit rate of the video stream $\beta(t)$ (cf., equation (2.7)).

$$\lambda(t) \approx \beta(t) \tag{2.7}$$

Without any further measurement, it is not possible at the endpoints to determine which router dropped a packet. On both ends of a connection, only the dropping rate of the complete path is measurable. The dropping probability of the path $w(t)$ depends on the dropping probability of each link $w_i(t)$ (cf., Figure 2.3) and can be calculated with equation (2.8).

$$w(t) = w_1(t) + w_1(t) * w_2(t) + ... + w_1(t) * w_2(t) * w_3(t) * ... * w_n(t) \tag{2.8}$$

The filling level variation of the queue on an outgoing port can be calculated with the incoming traffic and the capacity. The variation of the filling level f in an interval [t_1,t_2] can be calculated with equation (2.9).

$$f = \int_{t_1}^{t_2} (\beta(\tau) - \mu(\tau))\, d\tau \tag{2.9}$$

2.2.2 Estimating the Available Bandwidth

Measuring the available bandwidth is a well researched area and a lot of tools exist to measure it. The common approaches to measure the available bandwidth, such as PGM (Probe Gap Model) or PRM (Probe Rate Model), are independent of the application area and require extra probing packets [56]. 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 a good overview on these approaches including a comparison of common bandwidth estimation tools we refer to [20], [21]. These techniques are intrusive and rather slow. Therefore, they do not fulfill the requirements for a real-time video conference.

PGM sends probing packets with a small, fixed gap (time of transmission difference between two packets) between them over a path to measure the influences of potential side traffic. On a free 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 bit rate 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.

2.3 Interaction of Video Streams and Network Protocols

The video stream is sent via a transport protocol over the network and the video stream quality is influenced by the interaction of the network with the transport protocol. Multimedia applications use a variety of protocols that operate on different layers and their interaction needs to examined to allow a bandwidth aware video adaptation.

2.3.1 Network Protocols in Multimedia Applications

In general, the multimedia stream can be transmited via TCP or UDP. The preferred protocol depend on the use-case and the area of application. TCP is preferably used in web-based applications that use HTTP. TCP handles packet loss and retransmit the packets, which prevent decoding artifacts in the video stream, but can also lead to head-of-line blocking. Head-of-line blocking occurs, if the TCP experience packet loss or packet re-ordering and the TCP protocol waits until the required packet arrive. This leads to delayed frames. In real-time scenarios, it is more beneficial to drop these frames and procede with the next frame to avoid delays. In these situations, UDP is often used as transport protocol. UDP is an unreliable protocol and transmits data without a flow control, retransmissons, or QoS metrics. A common area of application is video conferencing, where artifacts in the video stream are less impairing than delayed video streams.

A very common protocol to transmit multimedia data, is the *Real-time Transport Protocol* (RTP) [51]. It focuses on multimedia transmissions and provides a standardized packet format for delivering audio and video data. It is often used in communication and entertainment systems that involve media streaming. Notable parameters that are contained in the RTP header are:

- The RTP version
- The type of payload (eg., the codec)
- A sequence number to identify the order of incoming packets and to detect packet loss
- The timestamp of the multimedia data, which is important for a correct playout.

- The Synchronization source (SSRC) field, which contains unique identifier to synchronize incoming packets with the right source.
- The Contributing source (CSRC) field, which is a list of SSRC identifier that contributed data to the RTP packet. For example, if multiple participants speak at the time, the RTP packets that contain the mixed audio data include their SSRCs in the CSRC list.

With the SSRC, the sequence number, and the timestamp, a receiver is able to decode a received packet of the multimedia stream correctly. RTP carries only media streams, but with the *RTP Control Protocol* (RTCP) [51] QoS metrics can be monitored. Usually, RTP is transmitted via UDP, which is unreliable and does not retransmit packets that are lost. Video impairments and especially the resulting QoE depend on the used protocol, the codec, the network conditions, and the loss rate. For an unreliable transmission, the factor with the most influence to the QoE is packet loss [8]. The decoder is not able to decode the video stream correctly if important frames are missing.

2.3.2 Affects of Packet Loss

The impairments of packet loss to the video quality are hard to predict and still need research [11]. The impairments depend on the loss rate, the codec and frames that are affected. Also, different loss pattern (with the same lossrate) have different influences on the video quality [7].

In this work, a reliable transmission protocol is used and thus, dropped packets are retransmitted. This is beneficial for the QoE, because no packets are lost and the decoder is always able to decode the video stream correctly. On the other hand, the retransmission causes a delay and also increases the overhead. This trade-off has to be done for each application and the application area individually. Nevertheless, packet dropping has huge effects for both video transmission approaches. Therefore, the effects of packet drops needs to be analyzed. The reasons for packet loss and their appearance time are often unknown and not predictable. Most common reasons are the following:

- One of the most common and most frequently reasons is the transmission over a wireless network, where packet loss occurs fairly often. This already caused a lot of trouble for older TCP implementations, where packet loss is used as indicator for a congestion and influences the window size [41]. Packet loss caused by natural influences in wireless network does not necessarily indicate a congested link and therefore, a downscaling of the sending rate might be unnecessary. Unfortunately, there is no easy way to determine if a link on a path is a wireless link [19].
- A rather unusual reason is packet loss caused by a broken link or malfunctioning or misconfigured router.

- Natural influences can also influence the reliability of a link. An insufficient shielded link might drop packets due to noise, which depends on the environmental influences. These are hardly predictable and might also change frequently.
- Common router AQM algorithms starts dropping packets before the router queue is completely filled. The dropping decision depends on the used AQM and is not predictable nor useful to determine the queue filling level of the router.

The various causes for packet loss makes it hard to determine the cause for packet loss. However, some approaches exist to detect packet losses caused by wireless networks. TCP had a lot of trouble in that regard, some TCP implementations try to detect wireless links [41]. In wireless networks, the use of packet loss for congestion detection is too unreliable. With a reliable transmission protocol, lost packets are retransmitted and no artifacts disturb the QoE. However, the packet loss influences the transmission time of the frames. Every packet loss causes a retransmission of the lost packet and also increases the congestion windowsize of the transmission protocol, if it supports congestion avoidance.

Packet loss is detected with a missing acknowledgement in a certain period of time, called retransmission timeout (RTO). The best known reliable transmission protocol is TCP. If an acknowledgement does not arrive in the RTO, the packet is retransmitted. The RTO is adapted to each connection individually and is calculated with the approximated smoothed transmission time (SRTT) and the variation of the round-trip-rit (RTT) [43].

A detected packet loss can also influences the sending behavior, because protocols like TCP use it as an indicator for congested router. TCP uses a sliding congestion window algorithm to avoid congestions. It contains the number of segments to sent in a window and is initialized with one maximum segment size (MMS). For every received acknowledgement, the congestion window is increased by one MMS. This exponential increasing stops when the slow-start threshold is exceeded. After that, the congestion window is increased by one if all packets in the window are acknowledged. This goes on until the requested windowsize from the receiver is reached. If packet loss occurs, the congestion window size is set to one and the slow-start threshold is set to half of the congestion windowsize.

Modern TCP implementations like TCP Veno take also the lossy characteristics of wireless networks into account [19]. The aim is to differentiate between packet loss caused by a low bandwidth and other causes. This can be done with an approximation of the expected TCP throughput and the actual throughput. If they differ, the bandwidth is low and packet loss can be interpreted as an impending congestion. Veno estimates the number of packets in the router and if it extends a certain threshold, Veno expects the path to be congested and packet loss is interpreted as a result of a congestion. In this case, the congestion windowsize will be decreased. If the queued packets is below the threshold, packet loss will be interpreted as a transmission error by a wireless network and will decrease the congestion windowsize only by a small factor.

For this work, we use the enet protocol, which is a reliable protocol and operates on top of UDP. It does not use a slow-start nor a congestion avoidance algorithm. In a video conferencing application, a congestion control is unwanted on the transport layer and instead the congestion control is done on the application layer. The A transport layer congestion control avoids congestions by lowering the sending rate, which is unwanted. The desired approach is to change the sending rate by scaling the video stream. With the enet protocol, packet loss only inflects the acknowledgement timeout window from enet. The higher the loss rate, the bigger is the acknowledgement timeout window and it takes longer until a lost packet is detected and retransmitted.

The retransmission of lost packets increases the transmission time for the whole frame. On a lossy path, the frame might be delayed at the receiver-side, which is perceived by the user as a stuttering video stream.

2.4 Adapting a Video Stream to the Network Conditions

Adapting a video stream to the network conditions is a wide research area. We will present and discuss approaches that already exist and focus on receiver-sided video adaptation (a), sender-sided video adaptation (b), hybrid approaches that use both sides (c), and HTTP streaming (d).

2.4.1 Receiver-sided approaches

Barzuza *et al.* presented TREND [4], a receiver-sided approach to scale a video stream based on the network conditions. TREND is designed for real-time video applications and does not use a buffer for the incoming video stream. Instead, the inter-arrival frame gap is used to detect a congested link. The TREND algorithm has two key functionalities: The delay detection and the bandwidth adaptation. The delay detection monitores the delay between received video frames on the receiver-side. A frame is considered delayed if the inter-arrival gap between two frames differs from the gap of the RTP timestamps. Basically, the TREND algorithm increases the quality of the video stream until a delayed frame is detected at the receiver-side. If a congestion is detected, the bandwidth adaptation set the video stream to the last stable bitrate before the congestion was detected and starts increasing it slowly after some time.

The additional delay between two frames is calculated with equation (3.1),

$$D_i = Rx_i - (Rx_{i-1} + \frac{TS_i - TS_{i-1}}{90})$$

where Rx_i is the receiving point in time of the RTP packet and TS is the timestamp of the frame. An exception are I-frames in this calculation, which are skipped. Their size is significantly bigger

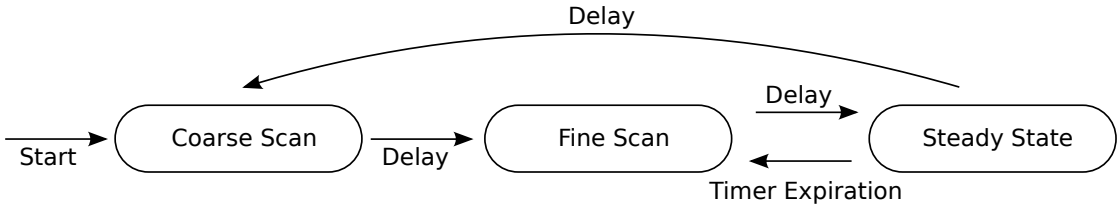


Figure 2.4: *State flow diagram of the TREND approach*

than P-frames and the additional transmission time increases the delay. This is undesired in the calculation, because the delay is not caused by a congestion.

The algorithm flow is show in Figure 2.4. It has basically two modes:

Scan Mode attempts to increase utilized bandwidth without exceeding the available bandwidth.

Steady Mode sends the video stream with a constant rate, while monitoring the available bandwidth for changes.

The algorithm starts with a coarse scan and the bit rate of the video stream is increased in big steps, until the receiver detects a delay. In this case, the current bitrate is dismissed and a fine scan starts with the previous, stable bit rate. The fine scale increases the bit rate in small steps until a delay is detected by the receiver. The algorithm enters the steady mode, where the video stream is sent with a constant bit rate and the receiver monitors the incoming packets to detect delays. If delays occur, the algorithm starts a new coarse scan. If no delays occur, the algorithm starts a new fine scale after some time to test if more bandwidth is available.

Another option to obtain information about the network conditions is to examine the video stream directly with an approach similar to PGM. Nguyen and Ostermann [40] present a scheme to scale a video stream at the receiver-side. Their work is part of a streaming system, which uses the scalability extension of H.264/AVC and provides a congestion control algorithm. Their approach is similar to the bandwidth estimation algorithm PTR, which is a PGM approach [25]. Instead of extra probing packets, the sending time of the RTP [51] video packets is manipulated for re-use 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 of the PTR tool.

2.4.2 Sender-sided approaches

Precise one-way available bandwidth estimations at the sender-side are complex and usually rely on the RTT or the loss rate, which are also influenced by the return path [27]. The measurement

results are less accurate than receiver-sided approaches, which are preferred for precise estimation. However, instead of a precise available bandwidth estimation, these metrics can be used as indicator for a congested link.

A sender-sided video adaptation approach that focuses on loss detection is presented by Euy-Doc Jang *et al.* [34]. They examine the influence of packet loss on a video stream and present an adaptation approach that indicates a congested path if packet loss occurs that disturbs the video transmission. A SVC video stream that consists of spatial, temporal, and quality layers is sent over a lossy path. Packet loss is caused by transmission errors or buffer overflow due to traffic congestion. Random packet loss leads to video streams, that are hard or impossible to decode for the decoder, especially if important frames are missing (eg., frames of the baselayer). The more frames rely on a missing frame, the lower is the quality at the receiver-side. This can lead to a very poor video quality, even if the incoming stream has a high bit rate. To solve this problem, a packet loss event is used as an indicator for a congestion. If a congestion is detected, the bit rate of the video stream is reduced by discarding less important packets with considering layer dependencies. This produces a higher decoding rate by an equal bit rate. A loss based congestion detection is not suitable for mobile video conferencing, because packet loss in a mobile realm does not always indicate a congestion, but is a common incident caused by natural influences. Another crucial requirement is the reaction time to a congestion. In a real-time video conference, it is necessary to scale the video stream before packet loss occurs. If packet loss occurs, the router queues are already filled. Depending on the router queue size, the video stream is already delayed at the receiver-side, which lowers the QoE. In an area of application, where the video stream is buffered at the receiver-side, this is not a problem, but for a real-time video conferencing software, a more direct and faster approach is desired.

G. Toma *et al.* [55], [15] presented a sender-sided approach, which indicates a congested link with variations in the RTT and use a scanning algorithm for upscaling. The general approach consists of an upscaling algorithm and a congestion indication algorithm that downscales the video stream if necessary. The application sends a video stream with a certain bit rate, which is the the lowest selectable bit rate in the beginning. For a safe upscaling, the probing algorithm temporarily increase the bit rate of the current video stream (burst) to emulate the next higher selectable bit rate. If the video stream is already close to the available bandwidth, the short burst will congest the link and the RTCP will feed-back a higher RTT. If the RTT does not change significantly, the higher bit rate is considered to be stable and the video stream will be sent with the higher bit rate. Additional probing packets are not required, because the RTP packets would have been sent anyway. The change of the bit rate is not noticeable for the user and does not influence the QoE, because of a buffer at the receiver-side. The drawback of this upscaling approach is that it requires a buffer at the receiver-side to absorb the bursts.

The downscaling algorithm does not need information about the any buffer states, but is based on the RTT deviation and the loss rate. If the RTT deviation or the loss rate exceeds a certain threshold, the path is considered to be congested and a video stream with a lower bit rate is

chosen. The approach was tested in controlled test setups and two real wireless networks. The results showed an advantage over non-adaptive streaming solutions. A disadvantage of this congestion detection approach is the slow feed-back time of the RTCP protocol. In the presented scenario, the receiving media players only sent a report every 5 seconds. For a video conferencing software, this approach is not suitable due to the required buffer at the receiver-side to absorb the burst of the upscaling testing.

2.4.3 Hybrid approaches

Jérôme Viéron and Christine Guillemot [59] presented a combination of a sender-sided and a receiver-sided approach. The basic bandwidth adaptation is based on the TFRC [23] protocol. The TFRC protocol however, does not take the characteristics of multimedia flows into account. Thus, they developed a new TCP-friendly protocol on top of RTP/RTCP, which takes the multimedia flows characteristics into account.

TCP-friendly protocols like TFRC can be classified into two main categories. Protocols that rely on an *Addaptive Increase Multiple Decrease* (AIMD) approach and react similar like TCP to packet loss [31]. This causes usually abrupt changes in the bit rate, which is not suitable for multimedia flows. The other category are protocols that are based on analytical models of the TCP throughput. In their work, they use both approaches. The throughput is estimated and also the loss rate is taken into account. The adaptation to the loss rate is similar to TFRC. However, most approaches assume constant packet sizes and TFRC recommends to set the maximum segment size (MSS) to an average. This penalize the multimedia flows when competing with TCP flows. Instead, they developed a new approach, which takes the varying packet sizes of multimedia flows into account. As result, the presented protocol shows a smoother rate variations than TFRC, which is beneficial for multimedia flows.

The network conditions and the available bandwidth are estimated at the receiver-side, while a video bit rate estimation is done at the sender-side and adapted to the estimated available bandwidth. A source rate control model scales a H.263+ compatible loss resilient encoder. Three approaches are proposed to map the estimated bandwidth to the encoder

1. The direct approach simply feeds the encoder with the available bandwidth as requested bit rate. It does not take the end-to-end delay or the encoder buffer status into account. This leads to severe timeout effects, which decreases the QoE for the user.
2. An approach with better results is provided when the encoder buffer state is taken into consideration. The resulting encoding bit rate must be able to drain the encoding buffer.
3. The best results are accomplished when in addition to the encoder buffer state also the delay constraints are taken into account, since estimated bandwidth does not include the

delays that are caused by buffering, decoding, and encoding. They lower the effective bit rate of the end-to-end transmission and are also considered.

The coupling of both rate adaptation approaches decreases the timeouts phenomenon significantly and minimize the expected distortion of the decoded stream. The approach is tested on a sample network, with controlled conditions but also via a large set of experiments on the Internet. The timeout effects are reduced significantly. The congestion control is compatible and also takes the characteristics of multimedia flows into account. The downside of this approach for our area of application is the complexity, the scaling of the bit rate on the transport layer, and the reaction time to a congestion. The adaptation to the available bit rate tries to find a fitting bit rate, that produces a stable video stream, but the throughput as a metric for a congestion does not react instantly to a congested path. The other indicator for a congestion is the packet loss, which usually occur when the queues are already filled.

2.4.4 HTTP Based Adaptation

Another area of applications with high bandwidth demands are streaming servers. These approaches usually use HTTP streaming [42] over TCP. Huang *et al.* [26] examine the video bit rate adaptation strategies of three popular video streaming services (Hulu, Netflix, and Vudu). These services face the problem that they operate via HTTP and their adaptation algorithms compete with the TCP congestion control algorithm. Choosing the right video stream bit rate is complex. In comparison to video conferencing application, they are allowed to buffer the video stream at the receiver-side. The focus differs from video conferencing applications, but the work gives insights into the problems that video streaming applications face when they have to compete with side-traffic.

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) [46] or the Strobe Media Playback, which uses the Open Source Media Framework (OSMF) [18]. These video players are specialized to play videos, which are hosted on websites and are delivered via a HTTP stream [42]. The server provides the same video with different bit rates. The streaming client is made aware of the available bit rates 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 bit rate. If the buffer is filled over a longer period of time, the client requests a video stream with a higher bit rate. In most scenarios, the video is pre-encoded with different bit rates, 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 [32], which simplifies the traversal of firewalls and NATs, but it already has a congestion control and might inflict the application layer congestion control [2]. Also, a buffer-based video delivery estimation is not desired in a real-time video application like a video conferencing software, because the buffering at the receiver-side increases delay in an undesirable fashion.

3 Video Codec Adaptation

Variable	Description
$\mu(t)$	Available bandwidth on a path
$\beta(t)$	Bit rate of the video stream
$\omega(t)$	Frame rate of the video
$k(t)$	video scaling factor
d	Delay caused by packet loss
y	physical transmission delay
$L(t)$	Filling level of a queue at time t
$q(t)$	Queuing delay of a packet on path
$r_i(x)$	Reception time of the i-th bit of frame x

Table 3.1: *Video adaptation notation*

A common approach to determine a congested path based on video stream mechanisms is to observe the filling level of the receive buffer. Unfortunately, this is not applicable in a real-time video conferencing application, where an unbuffered video output is desired. In the following, we discuss different aspects when designing solutions for adapting the video codec. Table 3.1 summarizes the notation that we use for deriving algorithms in the following.

3.1 Problem Description of Bandwidth Adaptation

In a video streaming application with an end-to-end connection, both sides can observe different network metrics, which can be used to detect a congested path. Thus, we need to clarify which end is used to detect a congestion and how they can be detected.

3.1.1 Observation 1—Who should control?

In the spirit of end-to-end connectivity, solutions should not be implemented at middleboxes but at the sender or the receiver. The receiver has no direct influence on encoding or the transmission

of the video flow. It has to request the sender to scale the video stream if it detects a congestion. Unlike the sender, the receiver has no information about the transmission time for the packets of a frame. Receiver-based approaches are slower and introduce overhead.

For a fast video adaptation, the sender should control the video coding directly, because the media source transmits the data to the receiver and thus defines the required bandwidth. This is an advantage over a receiver-side adaptation as it does not need an additional response channel to inform the sender about the measured network conditions.

3.1.2 Observation 2—Which measurement indicates congestion?

Congestion is indicated by queuing delays at routers. Measuring the delay should not introduce additional signaling but use data already available in the network stack. The network metrics visible at the application layer depend on the protocol to transmit the video stream. Typically, stateful transport protocols like TCP, RUDP, or RTP/RTCP provide inherent information about the transmission time, usually in terms of the RTT. Thus, gathering RTT is light-weight. However, a high RTT does not necessarily indicate a high congestion. The RTT sums the one-way delay from the sender to the receiver as well as from the receiver to the sender. A congestion on the return path influences the RTT even if it does not influence the video stream. This leads to incorrect reasoning of the delay in particular in case of highly asymmetric delays, which are visible in the Internet [37].

For video stream transmission, only the one-way delay from the source to the destination is important. The one-way delay is not easy to measure, though. It requires synchronized clocks [12]—a complex task. It also produces overhead, especially in large-scale multimedia applications with multiple participants.

3.1.3 Combining both observations

The solution space to adapt the video codec with respect to the network conditions is influenced by several requirements. Based on our observations there are the following design principles:

- The sender controls the video encoding, but the sender lacks precise information about the path to the receiver.
- The receiver can derive detailed insights into the path conditions between sender and receiver by directly analyzing the time variation and the size of subsequent arriving frames.
- Round-trip time measurements are directly provided by the network stack, but single RTT values do not reflect congestion on the path from the sender to the receiver.

Application requirements	Involved Party	
	Sender	Receiver
Very fast adaption, very low overhead	✓	✘
Very precise adaption	✘	✓
Very precise and fast adaption	✓	✓

Table 3.2: *Overview of application requirements and involved parties*

Depending on the video scenario a sender-based, a receiver-based, or combining both might be appropriate. We summarize our findings in Table 3.2.

3.1.4 Identification of a Congested Path

A network link congests when its traffic demands exceed the effective bandwidth $\mu(t)$. In this work, we assume $\mu(t)$ as effective *remaining* bandwidth after all side traffic has been subtracted, and consider the traffic stream of a controllable video of bitrate $\beta(t)$, only. Thus a path congests when $\beta(t) > \mu(t)$ and the goal is to find a scaling factor $k(t)$ for the video stream so that the bitrate matches the effective bandwidth: $\beta(t) * k(t) = \mu(t)$.

For scaling, it is not necessary to measure the available bandwidth $\mu(t)$ directly, nor to know the current video bit rate, but it suffices to estimate the bandwidth ratio $k(t)$.

$$k(t) = \frac{\mu(t)}{\beta(t)} \tag{3.1}$$

The challenge is to detect a congestion based on the given network metrics and to simultaneously extract a scaling factor $k(t)$ that steers the adaption of the video codec accordingly.

3.2 A Sender-sided Algorithm

The idea for a fast sender-sided video adaptation is to detect increasing queuing delays in the jitter variation of the RTT. A basic implementation of this sender-sided congestion detection was presented in our previous work [33]. We assume a near packet-wise feedback from the receiver that can be harvested from stateful transport. It is worth noting that RTP/RTCP feedback according to RFC 3550 is too slow, but rapid feedback mechanisms have been standardised in [44].

The RTT consists of the physical transmission delay y, which is approximately constant, and the queuing delays of the on-path routers $q(t)$

$$RTT(t) = y + q(t) \tag{3.2}$$

As long as the path is congestion-free, queuing delays remain stable and of little variation

$$RTT'(t) = q'(t) \approx 0 \tag{3.3}$$

While the RTT commonly rises and falls on a limited scale, its (signed) jitter alternates around zero with a small jitter variation close to zero. Every congestion, though, adds a significant queuing delay to the on-path delay that differs from regular RTT fluctuations. The jitter turns positive, causing a significant jump in its derivative. We use this jump in the jitter variation as a trigger for a video adaptation

$$RTT''(t) = q''(t) \gg 0 \tag{3.4}$$

In detail with every feedback from packet transmission, we monitor the second derivative of the RTT and interpret irregular (positive) jumps as early congestion indicators. The effects of a congestion to the RTT can be shown with a small sample stream. Figure 3.1 shows the size of the sent frames, the RTT, and the jitter variation of the RTT. The video stream is a square shaped stream that reaches from 4k byte frames to a burst of 10k byte frames. The packets are sent with a 66 ms gap between them, which is on par with a 15 fps video stream. The available bandwidth is set to 1 Mbit⁄s, which is exceeded by frames greater than 8250 bytes

$$\frac{8250bytes}{66ms} \approx 1Mbit/s \tag{3.5}$$

Whenever the video stream exceeds the available bandwidth, the RTT rises and the jitter variation makes a significant jump.

After a congestion is detected at a time t_c, the scaling factor $k(t_c)$ needs to be extracted for the codec adaptation. Consider the time interval [t_c,t_f] between the transmission of two frames. The queuing occurs at routers that have an egress to a congested link and the queue process follows the rate equation

$$L(t_f) - L(t_c) = \int_{t_c}^{t_f} \beta(\tau) - \mu(\tau)\, d\tau \tag{3.6}$$

where $L(t)$ is the filling level of the queue, t_c is the timestamp of the detected congestion, and t_f is the time when the next frame needs to be encoded.

The expected queuing delay $q(t)$ of a packet traversing at time t can be calculated as the ratio of the filling level and the departure rate from the queue $\mu(t)$, which we assume to be constant

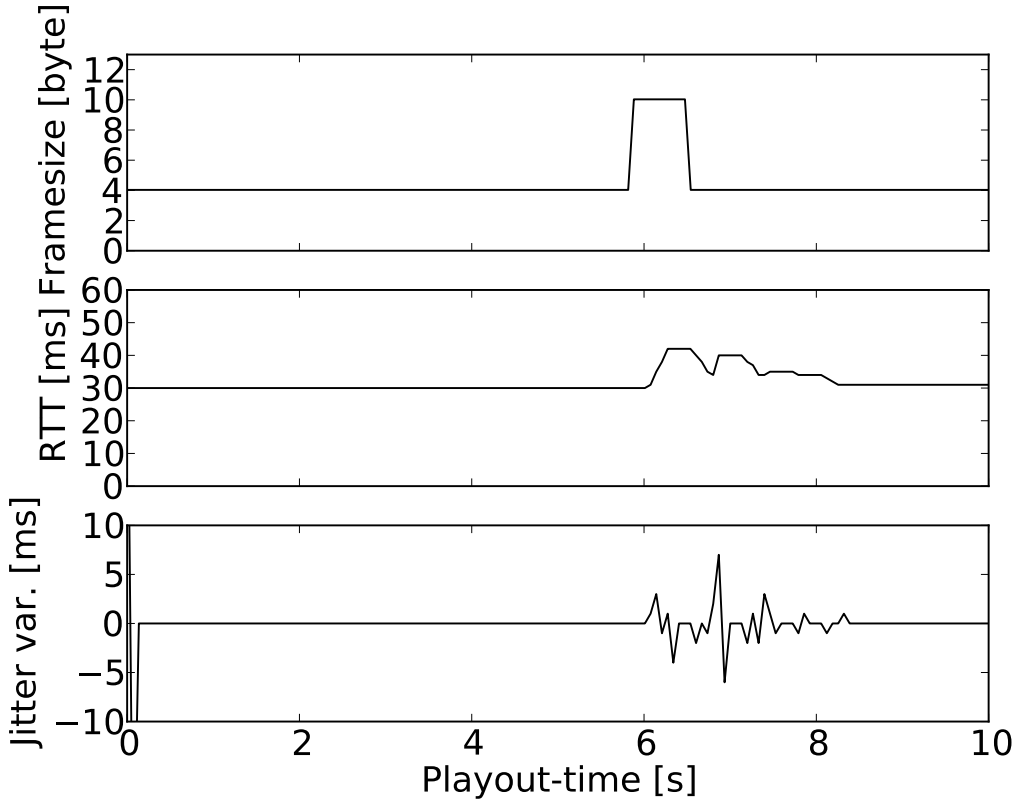


Figure 3.1: *Analysis of the sender-sided congestion indication*

in short time intervals. Correspondingly, we can approximate the additional queuing delay Δq generated during our inter-frame transmission time interval as

$$\Delta q(t_c, t_f) \approx \frac{1}{\mu(t_f)} \int_{t_c}^{t_f} (\beta(\tau) - \mu(\tau))\, d\tau \tag{3.7}$$

As shown in (3.3), the queuing delay variations are also visible in the RTT jitter

$$\Delta RTT(t_c, t_f) \equiv RTT(t_f) - RTT(t_c) \approx \Delta q(t_c, t_f) \tag{3.8}$$

For a small interval, we assume the bit rate and the available bandwidth constant $\mu(t) = \mu,\ \beta(t) = \beta$, which resolves (3.7) combined with (3.8) to

$$\Delta RTT(t_c, t_f) \approx \frac{1}{\mu}(\beta - \mu) * (t_f - t_c) \tag{3.9}$$

This expression can be solved for k

$$k = \frac{\mu}{\beta} = \frac{(t_f - t_c)}{\Delta RTT(t_c, t_f) + (t_f - t_c)} \tag{3.10}$$

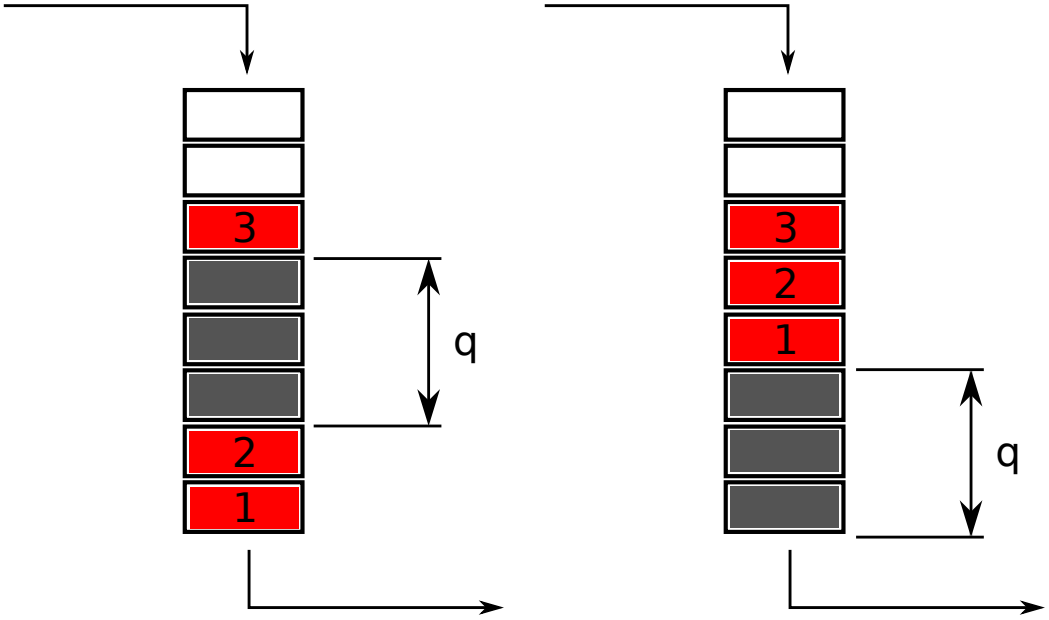
and remains with simple, measurable quantities known at the sender side. Still it remains unknown to the sender, whether the forward or the return path caused the congestion with additional queuing delay. However, assuming queuing delays always on the transmission path is a conservative approach and will never lead to a delay in adaptive scaling.

The sender-sided approach allows a very fast and light-weight codec scaling, which predicts impending congestions. It enables an immediate downscaling to assure a fluent video stream. On the downside, we only react to the jitter changes of the RTT, which do not provide information whether a link is currently free or congested. A decreasing RTT does imply that the video stream bit rate $\beta(t)$ is below the effective bandwidth $\mu(t)$, but we have no knowledge about the state of queues which still could be filled.

3.3 A Receiver-sided Algorithm

The sender-sided approach operates fast and provides a scaled video stream that stays below the available bandwidth. However, it has difficulties to identify a free link and optimize scaling for it. In case of a partially congested path, a receiver-sided adaptation can predict feasible video streams more reliably, as we will derive in the following.

On the receiver-side, several approaches are available to detect a congested path. In general, queuing delays influence the frame transmission time. Figure 3.2(a) shows such a scenario. A



(a) Packets of a video frame (red) are seperated by side-traffic packets (grey) in a router queue

(b) Packets of a video frame (red)are delayed by side-traffic packets (grey) in a router queue

Figure 3.2: *Queuing scenarios for a frame that consists of three packets*

frame, which consists of three packets (red) is queued in at a router and some side-traffic packets are also in the router queue (grey). They are located between the video packets and stretch the transmission time for the frame. This lowers the effective incoming throughput and we can detect queuing delay q if we measure this metric.

Measuring the incoming throughput for each frame, however, only detects queuing delays if the side-traffic stretches the transmission time for the frames. Figure 3.2(b) shows a scenario where the path is congested and queuing delays occur, but the transmission time of the frame does not change. In this scenario, no packets of the side-traffic are located between the packets of the frame in the router queue. To detect a congestion in this situation, the receiver can observe the time of reception for the whole frame. If it does not arrive in due time, delays occurred and the path might be congested.

In this application, both metrics are used to detect a congested link. They are also used to allow a safe upscaling if the path is userused. In the following, both approaches are described and compared to each other.

3.3.1 Measuring the Incoming Throughput

Detecting a congestion at the receiver-side can be done by comparing the incoming bit rate with the current bit rate of the video stream. If the incoming bit rate at the receiver-side stays below the bit rate of the video stream, a link is considered to be congested. Otherwise, whenever the incoming bit rate is higher than the origin bit rate of the video stream, the path is underused and video stream can be upscaled. Similar to the sender-sided approach, the factor k is used to scale the video codec, which represents the ratio of the available bandwidth μ to the video bit rate β.

The smallest and fastest accessible time interval to measure the sustained video bit rate is given by the inter-frame gap defined by the frame rate $\omega(t)$. The frame rate is known by the receiver and it can be used to calculate the current bit rate for a frame received with length l

$$\beta(t) = l * \omega(t). \tag{3.11}$$

For an approximation of the available bandwidth, the receiver continuously measures the incoming bit rate of the video stream, including the reception-time of the first and the last bit of a frame with n bits

$$\mu = \frac{l}{r_n(x) - r_1(x)} \tag{3.12}$$

where $r_i(x)$ is the reception-time for the i-th bit of frame x.

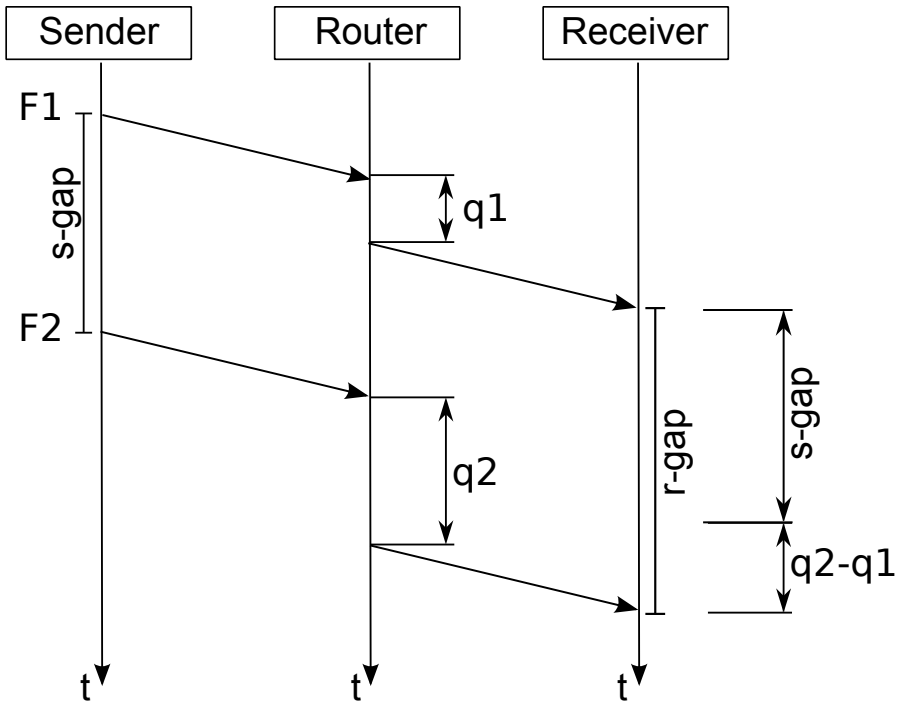


Figure 3.3: *Concept of detecting queuing delays in the inter-arrival jitter*

Combining equations (3.11) and (3.12) yields the scaling factor k as predicted from the receiver side

$$k = \frac{\mu}{\beta} = \frac{\omega}{r_n(x) - r_1(x)} \tag{3.13}$$

The presented receiver-sided approach differs from the sender-sided approach by making use of complete frames for the measurement. Therefore, it is slower, but it is capable to recognize an underused link and can suggest an accurate upscaling.

3.3.2 Detecting a Queuing Delay in the Inter-arrival Jitter of Frames

Another approach to detect a congestion at the receiver-side is to measure the variation of the frame inter-arrival gap. If the inter-arrival gap between two frames at the receiver-side differs from the origin gap they had at the sender-side, queuing occurred that stretched the inter-arrival gap.

The goal is to extract the scaling factor k from the increased receiver-sided gap r. The general

concept of inter-arrival gap measurement at the receiver-side is shown in Figure 3.3. It shows two frames F1 and F2 that delayed by the router. At the sender-side, they have a gap s between them that usually depends on the frame rate. Both frames get queued at the router and arrive at the receiver-side with a delay. The queuing delay of the second frame, however, is longer and stretches the inter-arrival gap at the receiver-side by $q2 - q1$. We want to detect these delay variations Δq is in the variation of the inter-arrival gap Δr that includes the gap variation from the sender-side Δs. If the frame rate is constant, Δs does not vary and is 0. Unfortunately, Δs also includes the encoding time variations. To extract Δq from Δr, Δs is required parameter at the receiver-side before the scaling factor k can be calculated.

At the sender-side, the gap s between the frames is measured when the frames are sent and is ideally given by the frame rate ($s = \frac{1}{\omega}$). In this application, s is not constant, but is influenced by variations of the encoding time

$$s_i = \frac{1}{\omega} + \Delta e_i \tag{3.14}$$

where Δe is the variation of the encoding time. With equation (3.14), the jitter of the sender-sided gap is

$$\Delta s = s_{i+1} - s_i \tag{3.15}$$

where i is the frame number. If the encoding variation is negligible, the receiver only needs to know the frame rate of the video.

The inter-frame gap r at the receiver-side is measured with the time of reception of the frames

$$r = r(x) - rx(x-1) \tag{3.16}$$

where $r(x)$ is the time of receiving the last bit of the x-th frame. With equation (3.16), the variation of the receiver-sided gap is

$$\Delta r = r_{i+1} - r_i \tag{3.17}$$

where i is the frame number.

In addition to the queuing delay variation Δq and the sender-sided gap variation Δs, Δr is also influenced variations of the frame sizes Δl. With equation (3.17) and (3.15), the inter-arrival gap Δr can be calculated with equation 3.18,

$$\Delta r = \Delta s + \Delta q + \frac{\Delta l}{c} \tag{3.18}$$

where c is the capacity of the path. The frame size variations need to be eliminated to extract Δq. The receiver has several options to eliminate the influence of the frame size variation.

1. Encoded video frames have often approximately equally sized frames as their neighbors, because they only transmit the changes to the frames they depend on. The inflects of the frame size variation can be reduced with observing an average Δr over a few frames. With the average Δr, we can react to the general trend of the inter-arrival jitter. This is a simple approach with a slow reaction time and a low accuracy.

2. The receiver can constantly measure the incoming bit rate of the video stream. It can be measured with the reception-time of the first bit and last bit of a frame (cf., equation (3.12)). It is not easy to determine the reception-time of the first bit. Operating system specific system-calls are necessary, which is possible, but a more portable and simple solution is to use the reception-time of the last bit of the first packet and the reception-time of the last bit of the last packet. With equation (3.19), the incoming rate μ can be calculated,

$$\mu = \frac{l - p_1}{rx_n - rx_1} \tag{3.19}$$

   where l is the size of the frame, p_1 is the size of the first packet, and rx_i is time of reception for a packet. This is an accurate approach, but also requires access to the reception time of the packets on the transport-layer.

3. The receiver ignores all significantly bigger or smaller frames to reduce the influences of the frame size variations. Only frames with similar sizes are considered for the measurement. This is a very simple solution, but also reduces the amount of frames, which are used for measurement.

The optimal solution depends on the area of application and on the desired accuracy of the measurement. For our application, the incoming rate at the receiver-side is already measured and the frame size variation delay can be eliminated. Without the frame size variation delay, Δq can be calculated with equation (3.20).

$$\Delta q = \Delta r + \Delta s \tag{3.20}$$

If the queuing delay does not change ($\Delta q = 0$), the gap at the sender-side and the gap at the receiver-side ideally do not differ ($\Delta r = \Delta s$). If Δq is positive, the queuing delays increased and if it exceeds a certain threshold, the path is congested.

The approach is tested with a small sample stream that is also used for the sender-sided approach in Section 3.2. The video stream is a square shaped stream that reaches from 4k byte frames to a burst of 10k byte frames and is sent with 15 frames per second. Figure 3.4 shows the size of the sent frames and the inter-arrival jitter variation. Whenever the video stream exceeds the available bandwidth (1 Mbit/s), the inter-arrival jitter shows a clear peak that differs from the normal inter-arrival jitter. In comparison with the sender-sided congestion control (cf., Figure 3.1), the receiver-sided approach detects the congestion a bit slower. The sender-sided approach shows

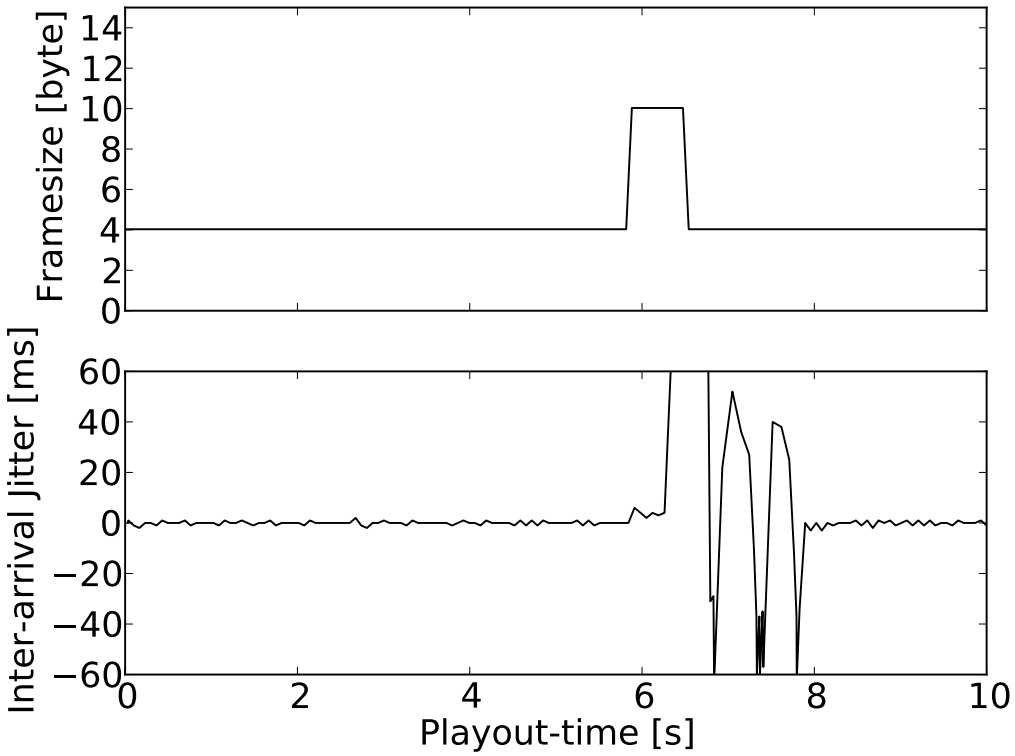


Figure 3.4: *Analysis of the receiver-sided congestion indication*

a significant peak in the jitter variation after 6 seconds, while the receiver-sided approach shows a significant peak in the inter-arrival after 6.4 seconds.

For the video adaptation, we need to determine the ratio of the available bit rate to the bit rate of the video stream with the measured Δq. Consider the measurement interval [t,$t+s$] between two frames, which correspond with Δs. With (2.9), we approximate the queue filling level variation

$$\Delta q(t, t+s) \approx \frac{1}{\mu(t_i)} \int_{t}^{t+s} (\beta(\tau) - \mu(\tau))\, d\tau \tag{3.21}$$

For a small interval [t,$t+s$], we assume the bit rate and the available bandwidth constant $\mu(t) = \mu,\ \beta(t) = \beta$, which leads to

$$\Delta q(t, t+s) \approx \frac{1}{\mu}(\beta - \mu) * (t+s) - t) \tag{3.22}$$

With the approximation of equation (3.21) in equation (3.22), the ratio k of the available bandwidth μ and the bit rate of the video stream β can also be determined by the ratio of Δr and Δs.

$$\Delta r = \Delta s + (\frac{\beta}{\mu} - 1) * \Delta s \tag{3.23}$$

$$k = \frac{\mu}{\beta} = \frac{\Delta s}{\Delta r} \tag{3.24}$$

The advantage of this approach is that it detects all queuing delays and it works even if no access to the time of reception of packets is granted. On the downside, it is not able to detect a free link. If both approaches are combined, the latter detects more impending congestions, while the former is able to suggest reasonable upscaling settings.

3.3.3 Effects of Retransmission Delays

In this work, a reliable transmission protocol is used and dropped packets are retransmitted, which increases the transmission time and the receiver-sided gap on packet loss. Therefore, both receiver-sided measurements suffer from imprecise measurement results if packet loss occurs. It is hard to predict, which packets are dropped and how long the retransmission will take (eg., they could be dropped again). The receiver can take that into account if it is possible for the receiver to determine the frames that experienced packet dropping.

The receiver has several options to handle frames, which experienced packet dropping. The receiver could take the delay from the retransmission into account for the calculation of the receiver-sided gap

$$d = \sum_{i=0}^{n} R_i * \mu * p_i \tag{3.25}$$

where n is the number of retransmitted packets, R is the amount of retransmissions and p is the size of the packet. Considering the retransmission timeout (RTO) delay before a packet is retransmitted, the equation (3.25) extends to

$$d = \sum_{i=0}^{n} R_i * \mu * p_i + R_i * RTO_i \tag{3.26}$$

The receiver must know the RTO for each packet and also the number of retransmissions. Also the effective bandwidth μ is necessary to calculate d with equation (3.26). A much simpler solution to this issue is to ignore all frames, which experienced packet dropping (if this information is available on the receiver side). This simplifies the calculation, but also reduces the amount of measurement frames on a lossy network path. The enet protocol does not provide this information at the receiver-side and therefore, we do not know how many packets of a frames experienced packet loss, nor do we know if a frame experienced packet loss at all. Without any further modifications, the receiver does not notice packet loss and we have to ignore delays caused by retransmissions. As a result, we expect inaccurate scaling suggestions on lossy paths.

3.4 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 resolve the congestion. The sender uses the jitter variation of the RTT as indicator for the congestion indication, 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 sender is only able to detect a increasing RTT and predict a congestion, but on a congested path, the sender loses the reference to the regular RTT fluctuations and cannot differentiate between a free and 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 congested. If the video stream is scaled with the jitter variation as indicator, it is necessary that it operates on a free path at the beginning and prevent every congestion. 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 inter-arrival gap between incoming frames is compared to the gap they are supposed to have. If they differ, the receiver sends a scaling request. In addition to that, the incoming throughput is compared to the supposed bit rate of the video stream. This metric is mostly used for upscaling suggestions, because the inter-arrival jitter congestion control reacts faster. 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 comparison to the sender-sided approach, the receiver is also able to detect an underused link.

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.

3.5 Parametrizing the Video Codec

The adaptation algorithms for video streams derived in the previous section lead to the scaling factor $k(t)$ that describes the ratio between the available bandwidth and a compliant video bit rate. Video codecs are not parametrized according to bitrate, but scale in the dimensions of quality, time and space. For our experimental evaluations in Section 5, but also for an application of the scheme in real-world applications, we need to translate scaling requirements into suitable codec parameters.

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 3.5. The graph shows the measured bit rate range for one of our test video with the highest quality (max. bit rate) and with the lowest quality (min. bit rate). The video source can only scale the video stream between these two extremes.

In this application, a high amount of temporal layer (higher frame rate) is preferred over a high quantization factor, because it leads to more measurement points for the congestion control. Therefore, we exchange a high quality, low frame rate video stream with a low quality, high frame rate video stream that uses more temporal layer, but try to keep the resulting bit rate the same. The challenge is to find the right exchange point. For multiple video sequences, the average bit rate for all possible encoding combinations are measured to find these exchange points. Figure 3.6 shows the average bit rate for every combination of the level of quantization and temporal layer. The resulting bit rate shows an exponential growth for an increasing quality, while each additional temporal layer shows a more linear increasing of the bit rate. All analyzed test sequences and show a similar behavior.

An additional layer is added, whenever an equal bit rate can be produced with more temporal layers but higher quantization factor. With the video analysis, the quantization factor 35 is determined as suitable exchange point. Figure 3.7 and Table 3.3 shows the full range of the level of

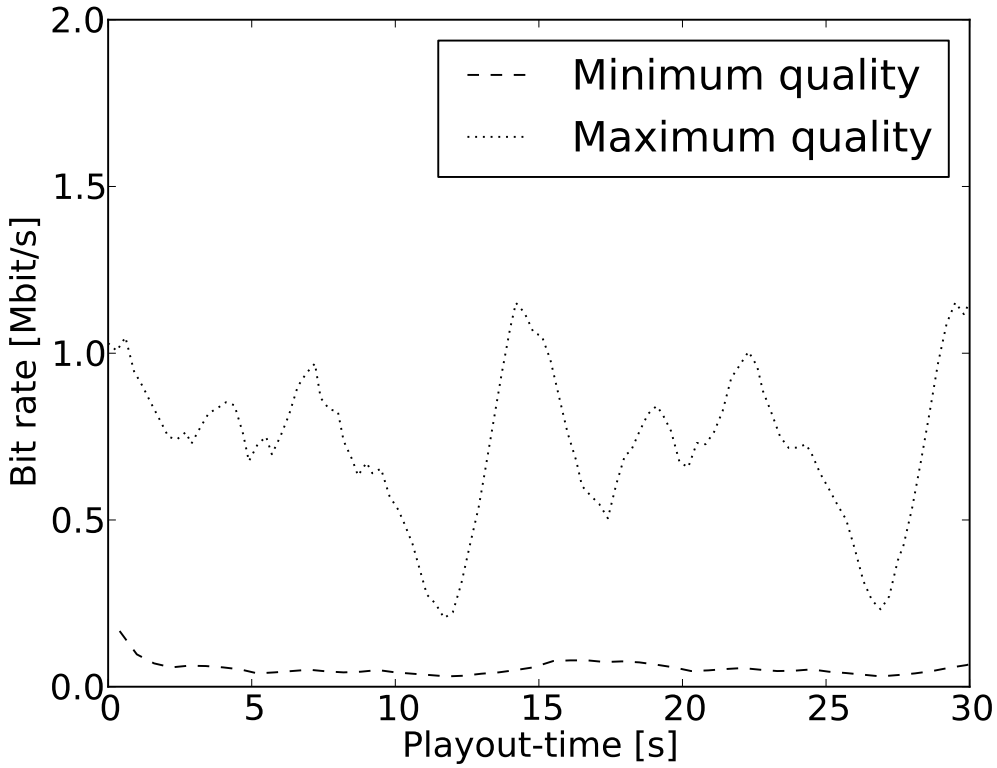


Figure 3.5: *Bit rate variation of a video stream*

Quality [%]	**Codec Settings**	
	Layer	Quantization
0-15	1	35...40
15-30	2	35...40
30-100	3	20...40

Table 3.3: *Quality and correspondent codec settings*

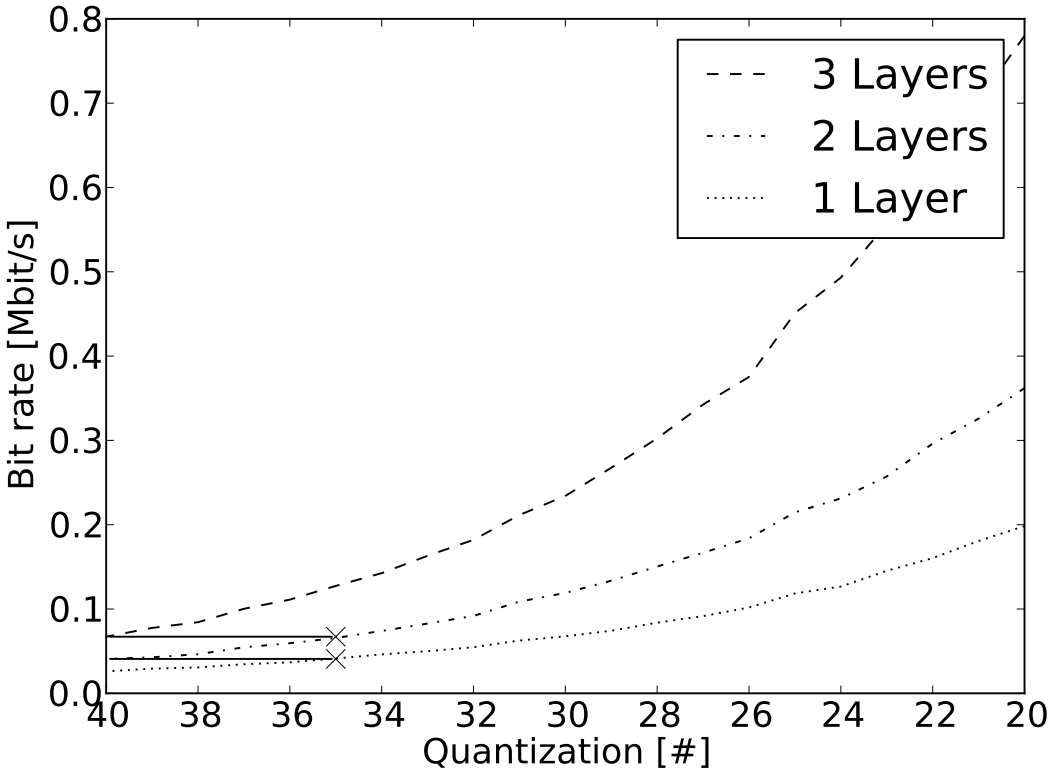


Figure 3.6: *Average bit rate for each encoding setting*

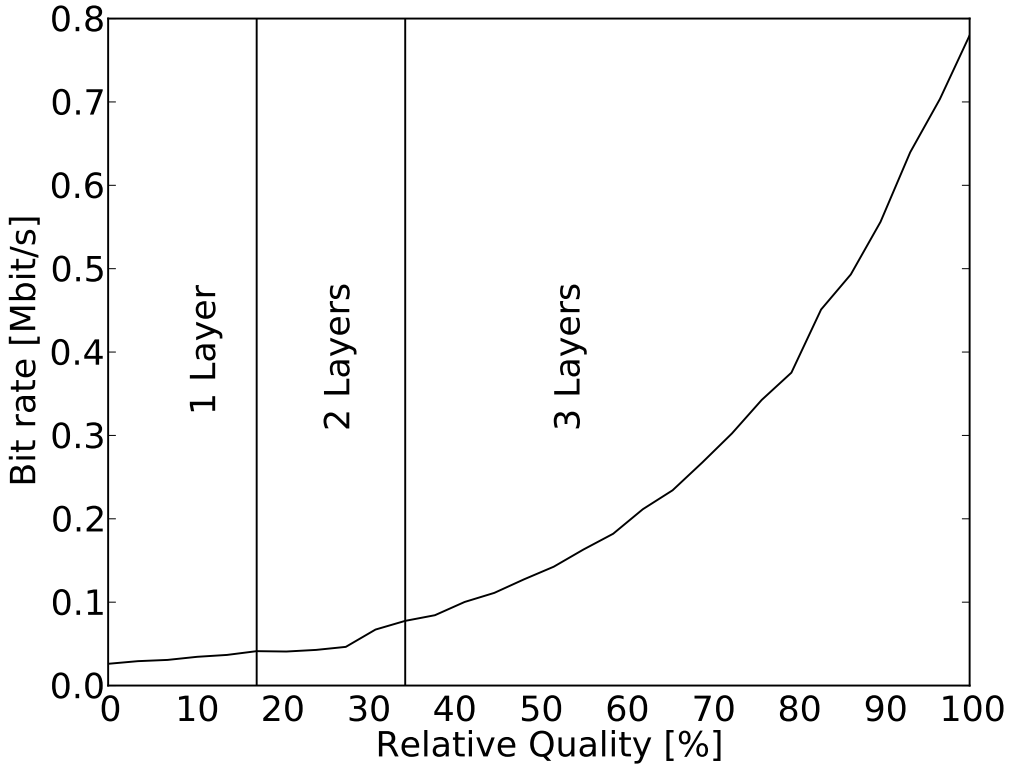


Figure 3.7: *Average bit rate of the testsequence "TW" for each quality setting*

quality settings in percent that can be used.

The congestion control provides a scaling factor k and the challenge is to feedback it into the codec adaptation. The measurements from the congestion control k needs to be mapped to the parameters the codec provides for scaling. Since the codec can only scale the bit rate $\beta(t)$ between the extremes β_{min} and β_{max}, it can be determined with equation (3.27),

$$\beta(t) = \beta_{min}(t) + \frac{Q(t)}{100\%} * (\beta_{max}(t) - \beta_{min}(t)) \tag{3.27}$$

where $Q(t)$ is the current encoding quality ($0\% - 100\%$). Correspondingly, the level of quality $Q(t)$ for a certain bit rate $\beta(t)$ is

$$Q(t) = \frac{(\beta(t) - \beta_{min}(t)) * 100\%}{\beta_{max}(t) - \beta_{min}(t))} \tag{3.28}$$

Based on the observed network condition, the congestion control suggests a relative scaling k for the current video stream. Suppose our video at time $t - \delta$ runs at bit rate $\beta(t - \delta)$ and gets rescaled such that $\beta(t) = \beta(t - \delta) * k(t)$. The old bit rate of the video stream $\beta(t - \delta)$ can either be measured or approximated with the old level of quality. The new quality of quantization can then be calculated for the current video stream bit rate as follows

$$Q(t) = \frac{(\beta(t - \delta) * k(t) - \beta_{min}(t)) * 100\%}{\beta_{max}(t) - \beta_{min}(t))} \tag{3.29}$$

With equation (3.29) and the mapping Table 3.3, the codec settings can be determined from the measured scaling factor k.

For example, let β_{max}=10 Mbit⁄s, β_{min}=5 Mbit⁄s and only the quantization factor is used for scaling. In this example, the mapping between the bit rate and the quality of quantization is linear for simplification and is shown in Table 3.4.

If the available bandwidth drops from 10 Mbit⁄s to9 Mbit⁄s, the congestion control will suggest a 10% bit rate reduction.

$$k = \frac{9Mbit/s}{10Mbit/s} = 0.9 \tag{3.30}$$

With the requested bit rate $\beta(t - \delta) * 0.9$ from equation (3.30)and equation (3.29), the new quality $Q(t)$ can be calculated

$$Q(t) = 80\% \tag{3.31}$$

The 10% bit rate reduction is archived with a 20% quality reduction. The new quality level is 80%, which produces a 9 Mbit⁄s video stream and matches the available bandwidth.

Quality [%]	Bit rate of the video [Mbit/s]
100	10
80	9
60	8
40	7
20	6
0	5
-	4
-	3
-	2
-	1
-	0

Table 3.4: *Quality mapped to the bit rate of a video stream*

Unfortunately, β_{min} and β_{max} are unknown, because they depend on the individual video stream. In order to scale the codec, we need at least a rough estimation of the data demands for the different quality settings of the video stream. Several precise approaches exist to estimate the bit rate of a video stream [53], but for our purpose, it is sufficient to use a simple exponential moving average for every frame with the highest or lowest encoding quality.

3.6 Scaling Timeouts and Quality Upscaling

The previous described approaches are responsible for the downscaling of the video stream, but we also face the following problems

- The video stream needs also to be upscaled
- We have competing scaling suggestions from the sender and from the receiver
- The effect of the scaling is not immediately visible, but takes some time until the measurement results arrive

In a measurement-interval $[t - \delta, t]$, the sender will provide downscaling suggestions, while the receiver provides downscaling and upscaling suggestions. Therefore, multiple scaling suggestions might arrive and we need some rules to prioritize them.

Downscaling request are always prioritized over upscaling request. If multiple downscaling requests are available, the lowest will be chosen. If only the sender-sided scaling is used, the upscaling is complicated, because we do not know the available bandwidth nor do we have a

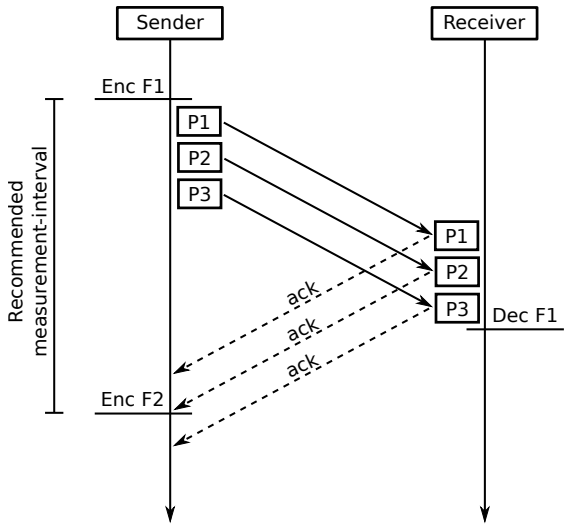


Figure 3.8: *Measurement results for one Frame F1, which is sent over the network via three packets P1,P2, and P3*

reliable indicator when more bandwidth gets available. The simplest approach is to increase the quality reservedly after some time without detected congestions and wait for the results of the congestion detection. If no congestion is detected, the new quality settings are suitable. If the receiver-sided approach is used for scaling, it is possible to detect an underused link if the incoming throughput is higher than the bit rate of the video stream. In this case, the receiver can suggest a video upscaling factor k. The sender collects all scaling suggestion in the measurement interval $[t-\delta,t]$. The recommended measurement interval is the inter-frame gap, because it is the longest period we can wait before the next frame needs to be encoded, for which the measurement results are needed. This is shown in Figure 3.8, where a frame F1 is encoded and sent over the network to the receiver. The frame F1 is split into 3 packets and the RTT is measured. Two of the three ack-packets are received before the next frame F2 needs to be encoded.

It is not beneficial to use a shorter measurement interval and therefore t_{min} in equation (3.32) is the minimum period of time, the sender should wait for scaling suggestions before he applies any scaling suggestions

$$t_{min} = \frac{1}{\omega(t)} \tag{3.32}$$

On network paths with a high latency, not all measurement results arrive in time before the next frame needs to be encoded (eg., P3 in Figure 3.8). In these situations, it is necessary to wait until the measurement for the current quality setting arrived before any new quality settings are applied. It takes at least the RTT to receive any measurement results of the currently applied

encoding settings. Since the RTT is not a constant value, the jitter should also be taken into account.

$$t_{min} = max(\frac{1}{\omega(t)}, RTT + |RTT'|) \tag{3.33}$$

To avoid too aggressive upscalings, an optional user timeout is respected in the upscaling timeout t_u and the equation (3.33) extends to

$$t_{min} = max(\frac{1}{\omega(t)}, RTT + |RTT'|, t_u) \tag{3.34}$$

4 Implementation of the Approach in a Multimedia Application

Video streaming in heterogenous network conditions is a complex task with high demands on the video streaming applications. To show that the presented approaches meet the previously discussed demands, the presented approach is implemented in a real-time video streaming application. Every instance can operate as receiver or sender that serves a video stream to multiple receivers. Table 4.1 lists the features the application provides. We focus on the measurement and analysis of the network metrics and is not meant as a productive video streaming software for end-user.

4.1 Video Streaming Application

The software is written in C/C++ and runs on Linux, Windows, and MacOS. The sender loads a video file or take a video stream from the webcam and send it encoded to the receivers. The open-source libav library[1] is used to load the video files, which allows a wide variety of multimedia formats and protocols. The default codec in the application is the DSVC codec [10], which is an extension of a very efficient H.264/AVC implementation. Optionally, the ffmpeg[2] library can also be used for enconding. The software provides a detailed and extendable graphical analysis of a wide set of network metrics. The measurement results are also written to log files in a csv-format for further analysis with extern tools.

The measurement settings can be loaded from configuration files and contain setting for the video files, the network conditions, and the scaling behavior. To change the network conditions on a path, a small additional application is implemented that runs on a router on the path. It receives the required network conditions and is able to manipulate the link conditions. Access to at least one of the router on the path is required to change the network conditions.

Instead of video files, it is also possible to generate video streams which do not contain a real video sequence, but produces certain video patterns. The generated video stream can be used to examine the reaction to specific combinations of network condition and video characteristics.

[1] http://libav.org/
[2] http://www.ffmpeg.org

Feature	Supported
Bandwidth adaptation of the video stream	✓
Graphical analyses of various network metrics	✓
Graphical analyses of video stream metrics	✓
Webcam support	✓
Various video formats and protocols	✓
Automated test scenario	✓
Various encoding codecs	✓
Network condition manipulation	✓
Easy exchange of the underlaying network.	✓
Generating automated testvideos	✓
Audio streaming	✘
Common MOS ratings	✘

Table 4.1: *Overview of the supported features*

The software is designed to independent of specific network protocols. For the sender-sided video adaptation, only a RTT measurement is required, while the receiver-sided approach only requires a response channel to inform the sender about the measured network conditions and the scaling suggestions.

4.2 Architecture of the Streaming Application

The streaming application consists of a sender and a receiver. Figure 4.1 shows the components of the application and their interaction. At the sender-side, a video stream is encoded via SVC and sent over the network. The source can be a webcam, a video file, or a generated test sequence. While the encoded video is transmitted, the RTT measurement is used by the congestion control algorithm at the sender-side to determine if the path is congested. The receiver also measures the network conditions, but instead of the RTT, it uses the inter-arrival gap of the frames and the incoming throughput to determine the network conditions. The measured network conditions and a scaling suggestion are sent back to the sender. With his own measurements and the measurement results from the receiver, the sender scaling component sets new parameters for the encoding in order to change the bit rate of the video stream accordingly to the network conditions.

Figure 4.2 shows sequence diagram of the application. The points in time when a new frame needs to be encoded and sent depends on the inter-frame gap (s-gap), which is given by the frame rate of the video source ($\frac{1}{\omega(t)}$). In a video conferencing software, the video source is

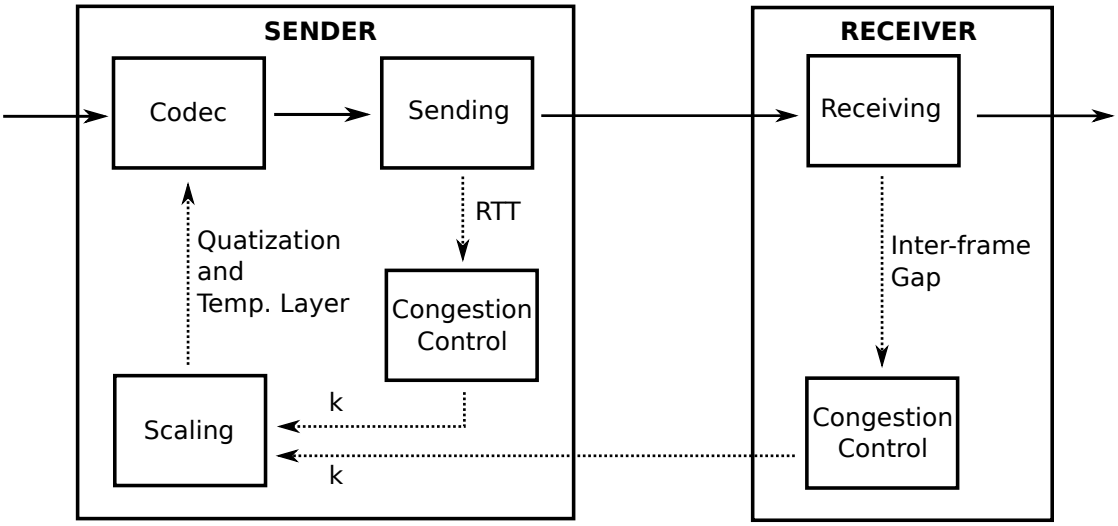


Figure 4.1: *Architecture of the application*

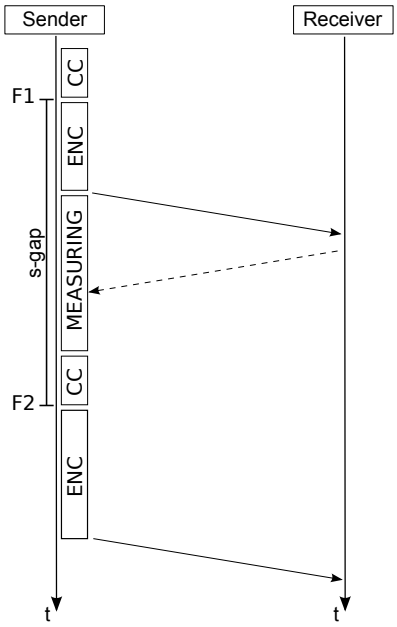


Figure 4.2: *Sequence diagram of the application*

usually a webcam with a constant frame rate. After a new picture is available, the frame needs to be encoded, which takes a certain amount of time that depends on the type of frame, the quality, and the processing power of the device. Thus, the encoding time can vary. The encoding quality and the resulting bit rate is determined by the congestion control (CC). At the beginning of a video stream, the encoding quality is usually low to prevent a congestion. Each sent frame is used for the measurement and the sender and the receiver provide a scaling suggestion. They are collected by the scaling algorithm at the sender-side and new codec settings are provided. The next frame is encoded with the new encoding settings.

In this scenario (cf., Figure 4.2), the measurement results arrive before the next frame needs to be encoded, which is not always the case. On a network path with a high RTT or if the encoding time is very high, the measurement results do not arrive in time an can only be considered for the following frame.

4.3 Parametrizing the Streaming Application

The application provides several parameters to set-up the adaptive video streaming. Depending on the network connection, the device, and the requirements for the video adaptation, the application can be parametrized with network specific settings, encoding settings and video adaptation behavior.

The network protocol in use (enet protocol) provides several settings that can be used to adapt the application to the network connection. Notable parameters are:

- Connection Timeout
- Reliably or unreliably transmission
- Maximum packet size
- Min./max. outgoing bandwidth
- Min./max. incoming bandwidth

In most scenarios, the network settings do not need to be changed.

Notable parameters that can be used to steer the codec are:

- The used codec
- Min./max. quantization factor
- Min./max. temporal layers

The parametrization of the codec influences the required processing power. For example, these settings can be used to optimize the software for mobile devices. In most scenarios, it is not necessary to change the encoding settings .

The parameters that are used to steer the video adaptation are:

- The upscaling timeout
- The downscaling timeout
- The threshold for the sender-sided congestion detection
- The threshold for the receiver-sided congestion detection

Theses parameters steer our video adaptation. The upscaling timeout is the time after an upscaling until a new upscaling is allowed. A high upscaling timeout out increases the time until a video stream reaches a high quality, but also decreases the likelihood to congest the link. The downscaling timeout does the same for downscaling suggestions. This can be usefull to prevent multiple downscaling suggestion in a short period of time. This avoids very aggressive downscalings and provides a more stable video stream, but increases the likelihood of a network congestion. The threshold for the sender-sided congestion detection is the RTT jitter variation that indicates a congested link. By default, this is set to a very fine-granular value of 5 ms. The threshold for the receiver-sided congestion detection is the value for the inter-arrival jitter that indicates a congestion. By default, this is set to a very fine-granular value of 5 ms. With the parameters of the video adaptation, the application can be optimized for different network conditions.

5 Performance Evaluation

The presented application is tested in multiple different network conditions. In this chapter, we will discuss the testbed that is used and analyze the measurement results.

5.1 Measurement Setup

The Internet consists of many heterogeneous links that transmits multiple competing traffic streams from different applications with different protocols. Thus, the network conditions are hard to predict and a lot of parameters are unknown. For testing purposes, a testbed with controlled network conditions is needed to examine the scaling approaches. For a detailed analysis, we need

- Controlled network conditions
- Sender with enough CPU power to encode a video stream in real-time
- Multiple clients
- Access to the router on the path

It is worth mentioning that the experiments cannot be conducted in the PlanetLab testbed [45] for two reasons. First, PlanetLab nodes provide too much bandwidth. Even though network links between PlanetLab nodes are very heterogeneous [39], our tests showed that capacities are still sufficient for typical video test sequences. Second, PlanetLab nodes do not provide sufficient CPU resources. To increase the required bandwidth, we could deploy HD video sequences. However, CPU resources of PlanetLab nodes are too low for appropriate encoding.

5.1.1 Emulation environment

For most measurements, we use Mininet[3]. Mininet is an open source network emulator, which can be used to create virtual networks with controlled network conditions. With mininet, we are able to emulate various scenarios, but it also has some downsides. The purpose of the testbed is

[3] http://mininet.org

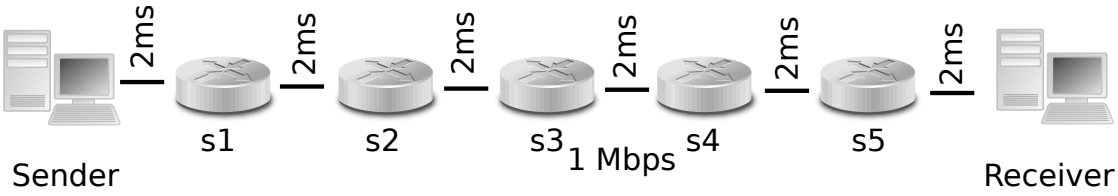


Figure 5.1: *Daisy-chain topology*

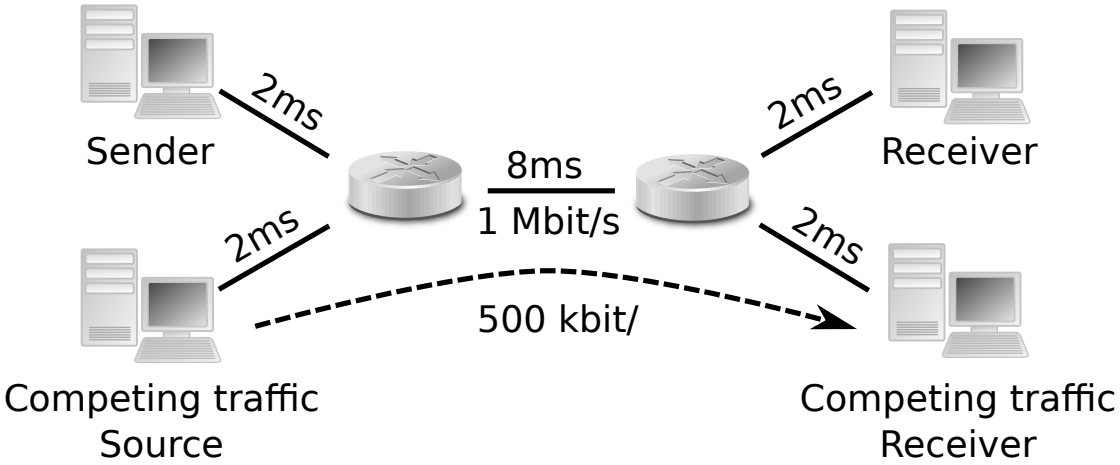


Figure 5.2: *Dumbbell topology*

to simulate a bottleneck on a link, but the conditions on the testbed differ from real networks like the Internet. It is difficult to emulate a 'normal' Internet link, because they are very multifaceted. The available bandwidth depends on the amount of UDP connections, TCP connections, and traffic characteristics of the competing applications. For example, a HTTP application such as a browser has short TCP peaks when a new site is requested, while a streaming application exhibits a relatively constant UDP stream. The applications might also react to a congested path and adapt its traffic stream to the available bandwidth. Emulating this variety on a very fine-grained level is difficult and usually does not help to highlight specific protocol effects. This variety is complex to emulate and in this work, we will focus on *basic* parameters such as the available bandwidth and the delay of a path in a representative emulation environment. To gain ground truth in our results, we also verified the measurements by running selected experiments in a real network.

5.1.2 Topologies

We used two basic network topologies in our performance evaluation, a daisy-chain topology (cf., Figure 5.1) and a dumbbell topology (cf., Figure 5.2). The daisy-chain topology consists of one sender, one receiver, and five interconnecting switches. The dumbbell topology consists of one sender, one receiver, two switches, and two hosts that produce a competing traffic stream. The competing traffic is emulated by iperf[4].

In both topologies, the bandwidth is by default limited to 1 Mbit/s, and the RTT between sender and receiver is set to 24 ms. This is considered to be a short RTT network [22]. The one-way delay for all (full-duplex) links has been adjusted accordingly, i.e., either 2 ms or 8 ms (cf., Figure 5.2). These are very friendly network conditions, because the fastest possible response time for the congestion detection is defined by the RTT (i.e., 24 ms). The gap between arriving frames is 66 ms for a video with 15 fps. This means that the signaling between sender and receiver can be completed before the next frame is encoded.

It is necessary to emulate a bottleneck on the path to limit the available bandwidth on the path. In real networks, the Linux program "tc" can be used to add traffic shaping rules.

When the bottleneck is set to a low available bandwidth, the test results showed huge variations in the transmission time for the incoming video frames. These are artifacts of the traffic shaping algorithm, which shapes the traffic by delaying Ethernet frames. The transmission time for the video frames vary a lot, because after every Ethernet-frame (which are considered to be full-sized Ethernet-frames), a 12.14 ms gap is inserted to archive the limit of 1 Mbit/s (cf. equation (5.1)).

$$1518Byte * 12ms \approx 1Mbit/s \tag{5.1}$$

This leads to very rough shaping artifacts in the incoming traffic streams. To minimize these influences, we set the bottleneck always to 10 Mbit/s and use a 9 Mbit/s UDP side-traffic stream to limit the available bandwidth to 1 Mbit/s. In some cases, the network settings are slightly changed to examine the adaptation in different network conditions. In the following, we will discusses several typical network conditions that needs to be examined:

Short-RTT paths The software is tested in a friendly network conditions with a low RTT and a very low loss rate. The used topology is the daisy-chain topology, which is shown in Figure 5.1. We are especially interested in how the sender alone, the receiver alone, and a combined approach react to different bandwidth limitations. A detailed analysis is given in Section 5.5.

Lossy path The loss rate of a path influences the transmission time for the frames and influences the bandwidth adaptation algorithm at the receiver-side. The sender-sided algorithm

[4] http://iperf.sourceforge.net

is not inflicted, because it uses only the RTT from successfully transmitted packages for the congestion detection. A detailed analysis is given in Section 5.7.

Long-RTT paths The effectiveness of the codec adaptation depends on the response time of the measurement. Therefore, a high RTT slows down the reaction time of the video adaptation and needs to be examined. This is tested on the daisy-chain topology, but the RTT between the switches is set to 192 ms and the RTT of the path is 200 ms. This is considered to be a long RTT network [22]. A detailed analysis is given in Section 5.8.

Effects of side-traffic The previous tests emulate a congested link by reducing the available bandwidth. To examine the effects of changing bandwidth limitations, the available bandwidth will be altered by side-traffic. For this scenario a dumbbell topology is used, which is shown in Figure 5.2. It consists of 4 hosts and 2 switches and the bandwidth is set to 1 $\frac{\text{Mbit}}{\text{s}}$ by default. In comparison to the daisy-chain topology, an additional side-traffic stream is sent across the path. A detailed analysis is given in Section 5.9.

Video Adaptation in a network with a congested return path The sender-sided approach uses the RTT for the congestion detection, which also includes the return path. If the return path is congested, the unidirectional video stream is not influenced, but the RTT increases. Section 5.10 examines this effect.

Video adaptation in a network with a constant increasing congestion The sender-sided approach uses a significant jump in the variation of the RTT jitter as indicator for a congestion. If the RTT increases slowly and with a constant rate, the sender-sided approach is not able to detect the congestion. This problem will be discussed in Section 5.11.

RTT variations A common characteristic of wireless network is a fluctuating RTT. This influences the sender-sided approach directly, but also influence the the receiver-sided approach due to the increased transmission time for frames. The effects are examined in Section 5.12.

Long-time test The previous tests use short test sequences and short measurement periods. This allows an accurate analysis of the measurement results. The software is also tested in a long-time test to the effectivity and reliability of the used approach over a long period of time. A detailed analysis is given in Section 5.13.

Real network The software is also tested in real networks to verify the effectiveness in uncontrolled network conditions. In this test, the exact available bandwidth, the RTT, and the loss rate are unknown and needs to be measured. A detailed analysis is given in Section 5.14.

5.1.3 Video sequences and video codec

For the measurement, we use the TW, UH, G4, SM, TC, and KO test sequences from the Heinrich-Hertz Institute (HHI). The resolution is 768x576 pixel and the frame rate is 15 fps. The

videos are too short for longer measurements. To achieve a total playout time of 60 s, we looped each test video. In general, the improvements based on the adaptive video coding were qualitatively very similar. For visibility reasons, most measurements uses the video sequence "TW". In Section 5.6 we show a comparison of all test sequences.

All measurements are conducted using the DSVC codec [10], which supports up to three temporal layers. It is an extension of a very efficient H.264/AVC implementation. The codec is used in commercial products and thus complies with real-world requirements.

The "TW" video sequence (cf., Figure 5.3) and its bandwidth characteristic are shown in Figure 5.4. 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 0.5 Mbit/s and 1.7 Mbit/s and it is feasible to transmit the video on a path with 1.7 Mbit/s available bandwidth without congesting the link.

5.1.4 Rating Metric

For our video conferencing application, a suitable QoE metric is needed, to make the measurements for the different network scenarios comparable to each other. In a multimedia application, the video stream impairments are usually caused by packet loss on the network layer, which leads to poor QoE ratings and the common QoE metrics focus on this problem. In our video conferencing software, a reliable transport protocol is used and the packet loss does not cause decoding artifacts. Instead, lost packets are retransmitted and the additional delay can lower the QoE [58]. Therefore, the most common QoE metrics are not suitable for our application, because they do focus too heavy on packet loss.

The best fitting rating metric for our purpose is the DF rating from the MDI. The MDI has the downside that it calculates the DF with the filling level of the buffer. In this work, a buffer is unwanted to fulfill the real-time requirements. Instead, the DF is calculated with the inter-arrival frame variation. The inter-frame gap from the sender-side is compared to the inter-frame gap from the receiver side for each frame of the video stream (cf., equation (5.2)).

$$j(i) = \Delta r(i-1, i) - \Delta s(i-1, i) \tag{5.2}$$

When a frame is heavily delayed and the following frame already arrived, $\Delta r(i-1, i)$ is zero. Therefore, the maximum negative inter-arrival jitter is determined by the sender-gap. The sender-gap is not constant and experience variations due to the encoding delays, but it jitters around the inter-frame gap given by the source of the video ($\frac{1}{\omega}$). The gap at the receiver-side $\Delta r(i-1, i)$ cannot be below zero, because the frames are reordered at the receiver-side. If one frame is delayed and the following frame arrives at the same time or even earlier, it gets queued at the receiver-side and the gap between the frames will be zero. Thus, the minimum jitter $j(i)$ is $\Delta s(i-1, i)$.

Figure 5.3: *One picture of the testsequence TW*

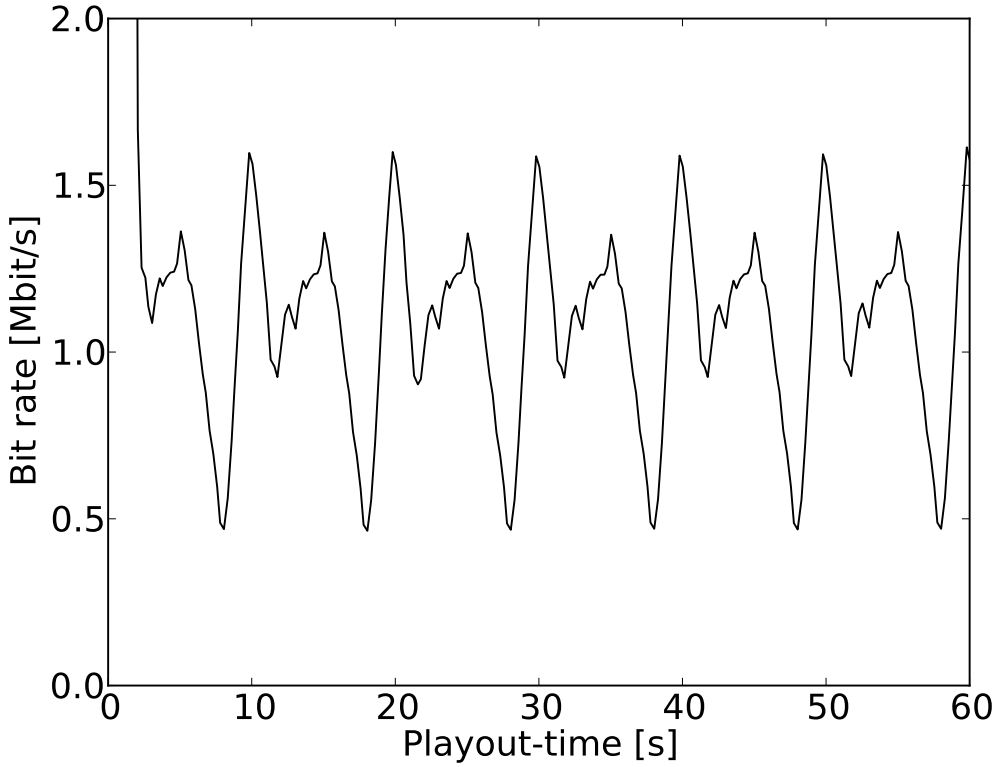


Figure 5.4: *Bit rate of the video stream*

The lower the jitter $j(i)$ is, the lower is the frame rate variation and the higher is the QoE. For example, on a ideal network with no influences on the video stream, the inter-frame gap on the sender-side and on the receiver-side would be equal and the jitter would be zero.

MDI defines a jitter below 9 ms as a perfect video stream and a jitter above 50 ms as unacceptable. To rate the video stream, the average over the playout-time could be calculated. The upside of this rating metric for the QoE is, that it is hard to classify videos from a human perspective. Instead, we count the 'good' frames that arrive with an inter-arrival jitter below 9 ms and use the ratio to the remaining 'bad' frames as a rating metric.

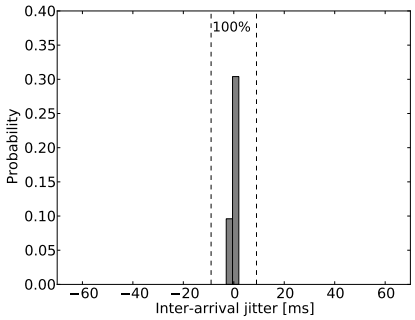
Another problem with this metric for our purpose is the assumption of a buffer at the receiver-side. It rates a high positive jitter as problematic, because it drains the buffer and the resulting underflow causes a stuttering video stream. A high negative jitter is rated as 'bad', because it indicates an impending overflow of the buffer. In our case, a negative jitter is not necessarily 'bad', because it usually is a counterpart to a previous delayed frame. It is a side-effect if a delayed video stream normalizes and evens out the delayed frames. Nevertheless, it decreases the QoE for the user when multiple frames are played out faster and with a wrong frame rate.

We evaluate the performance of the adaptation approaches based on the distribution of inter-arrival jitter. To allow for comparisons between the different experiments, the binning of the inter-arrival jitter is constant. We choose a width of two. Figure 5.5(a) shows the distribution of the incoming frame times on a free network path. Almost all frames arrive in due time and the gap between two frames on the receiver-side does not differ from the gap they had when sent. This is the result we are aiming for.

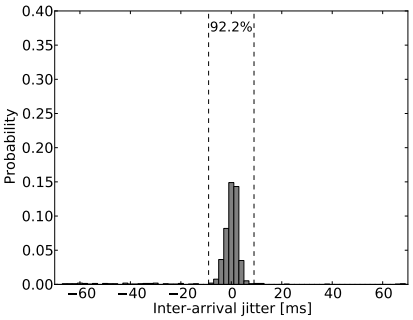
Figure 5.5(b), 5.5(c), and 5.5(d) shows the distribution of the inter-arrival variations on a path with 1.3 Mbit⁄s, 1 Mbit⁄s, and 0.7 Mbit⁄s. For the bottlenecks with 1 Mbit⁄s and 1.3 Mbit⁄s, the negative inter-arrival jitter shows a concentration point around -66 ms, which is the minimum jitter and depends on the inter-frame gap on the sender-side $\Delta s(i-1, i)$. For the positive inter-arrival jitter, no clear point of concentration exist and they are distributed over a wide range.

A comparison of the unscaled video stream for various bottleneck settings is shown in Figure 5.6. With less available bandwidth, the amount of frames that arrive with a inter-arrival jitter below 9 ms shrinks and the QoE gets worse. In this work, we only differ between frames that arrives 'perfect' (with a jitter below 9 ms) and frames that do not arrive in time. The higher the amount of frames with a 9 ms inter-arrival jitter or less, the higher the overall video quality.

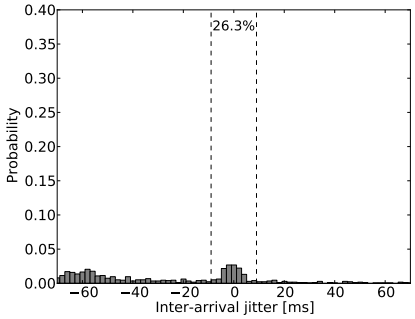
The 9 ms inter-arrival jitter from the MDI is a fitting metric to rate the applicability in a real-world deployment. The downside of this rating is that it only takes the inter-frame delays into account for the QoE rating. A video with a good rating does not necessarily have a high overall MOS rating. In our area of application, a delay-based rating is a suitable rating, because our focus lies on video conferencing, where an in due time arrival has a higher priority than a high video quality.



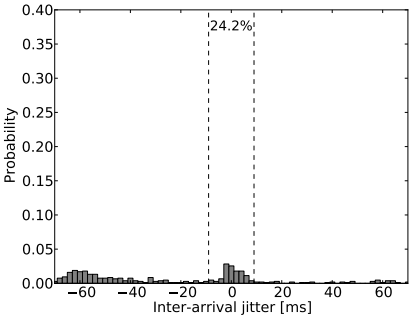
(a) Inter-arrival jitter measured on a 100 Mbit/s path



(b) Inter-arrival jitter measured on a 1.3 Mbit/s path



(c) Inter-arrival jitter measured on a 1 Mbit/s path



(d) Inter-arrival jitter measured on a 0.7 Mbit/s path

Figure 5.5: *Inter-arrival jitter distribution measured for different bandwidth limitations*

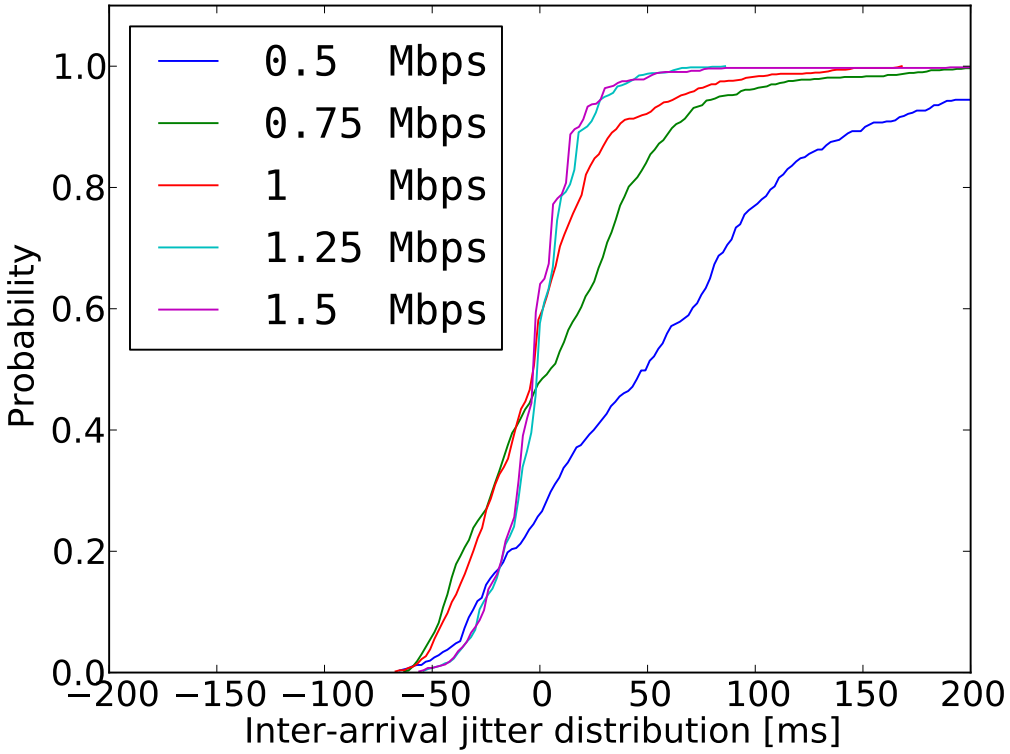
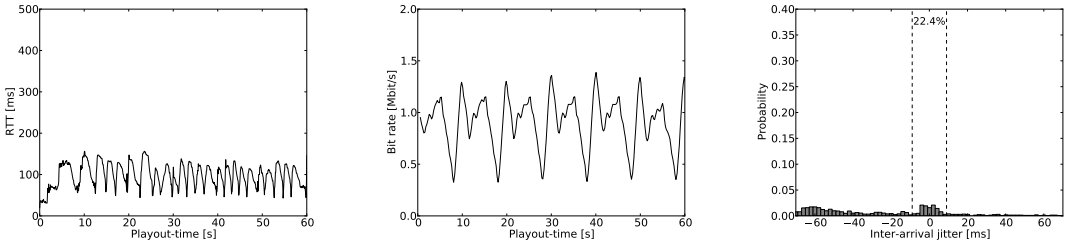


Figure 5.6: *CDF comparison for several bottleneck settings*



(a) RTT measured at the sender (b) Bit rate of the video stream (c) Inter-arrival jitter measured at the receiver

Figure 5.7: *Unscaled video stream with a maximum bandwidth of 1 Mbit/s*

5.2 Unscaled Video Stream

The video source sends an *unscaled* version of the TW test sequence with the highest quality, i.e., a minimum quantization factor is configured and all three temporal layers are enabled. The bit rate of the unscaled video stream varies between 0.5 Mbit/s and 1.5 Mbit/s (cf., Figure 5.7(b)). The available bandwidth of 1 Mbit/s is clearly exceeded. The resulting congestion influences the RTT significantly (cf., Figure 5.7(a)). In the best case, the RTT should fluctuate around 24 ms, but the overused link causes RTT variations between 24 ms and 500 ms. In a video conferencing application, the maximal one-way delay should be around 100 ms to not distract end users [28].

As a consequence of the congestion, frames do not arrive in due time. Figure 5.7(c) shows the distribution of the inter-arrival jitter per frame at the receiver-side. The huge variations in the inter-arrival jitter are perceived by the user as a stuttering video stream. The Media Delivery Index (MDI) [60] suggests an inter-arrival jitter below 50 ms for an acceptable video stream and an inter-arrival jitter below 9 ms for a high quality video streaming. Only 22.4 % of the frames fulfill these requirements; appropriate video conferencing is not possible.

We encoded the video stream with all variations of encoding settings to get a general idea of the optimal encoding settings for this scenario. The optimal encoding quality for this scenario is shown in Figure 5.8. It shows the bitrate for the video sequence "TW" if every frame is encoded with setting that provides the closest bit rate to 1 Mbit/s. The encoding settings vary between 70% and 100% and the resulting bit rate never exceeds the 1 Mbit/s available bandwidth. The goal is to match the pattern of this graph with the presented video adaptation algorithm.

In the next sections, we show how an adaptive video codec can cope with congestions. Our objective is to provide a fluent video stream.

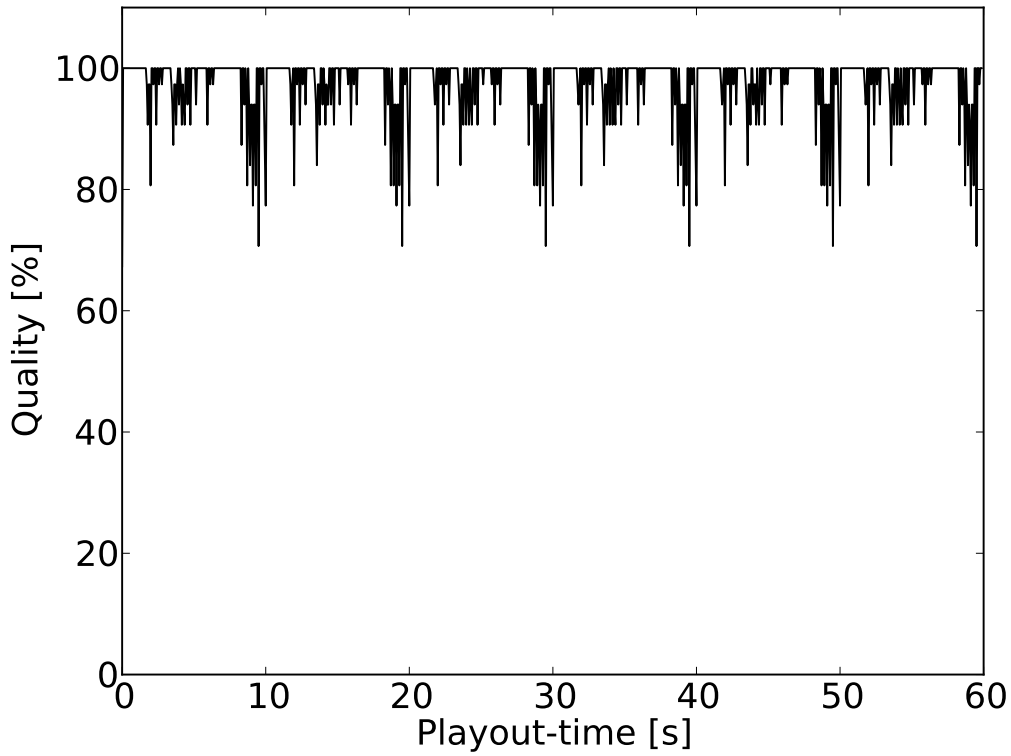
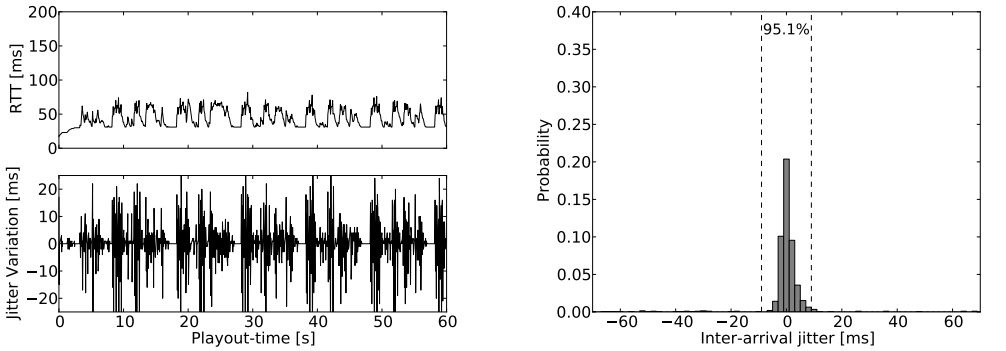


Figure 5.8: *Encoding quality for an optimal bandwidth utilization on a 1 Mbit/s path*



(a) RTT and jitter measured at the sender

(b) Inter-arrival jitter measured at the receiver

Figure 5.9: *RTT and inter-arrival jitter distribution of a sender-sided video adaptation*

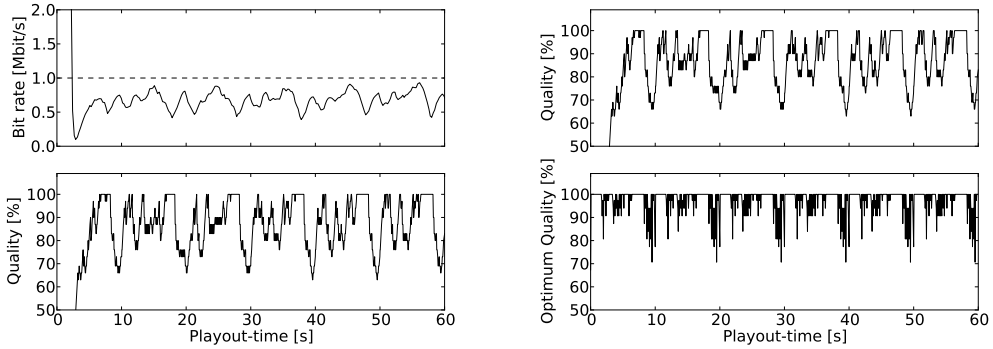
5.3 Sender-sided Video Adaptation

In this scenario, only the sender is used to scale the video stream without support from the receiver. The sender itself is not capable to detect a priori an uncongested path and therefore has to increase the video quality after some time when no congestion is detected. Until the sender explores congestion, we use a very aggressive upscaling strategy and continously increase the quality after a frame was sent.

Figure 5.9(a) shows the RTT and the jitter variation of the RTT, which is used to indicate a congested link. Every increasing of the RTT goes along with a significant jump in the jitter variation, which triggers a downscaling. Figure 5.9(b) shows the inter-arrival jitter distribution of the frame reception at the receiver-side. Compared to the unscaled video stream, 95.1% of the frames arrive in due time and stay below a 9 ms inter-arrival jitter.

Figure 5.10(a) shows the resulting bit rate of the video stream and the codec quality adaptation. At the beginning, the video stream starts with the lowest quality settings, but increases rapidly. Every detected congestion that increases the RTT is followed by an adaptation to the estimated available bandwidth. The encoding quality varies between 75% and 100%, which is still a high quality. The resulting bit rate of the video stream stays below the available bandwidth and does not congest the link.

The encoding quality is also compared to the optimal encoding settings and is shown in Figure 5.10(b). The optimal encoding settings starts with 100%, while our approach starts a the lowest quality settings and increases until the operating point is reached. Besides the faster up and downscaling of the optimal encoding settings, the two graphs show similar characteristics.



(a) Bitrate and codec adaptation of the video

(b) Comparison of the encoding quality suggestion with the optimal encoding quality

Figure 5.10: *Bitrate analysis of a sender-sided video adaptation*

5.4 Receiver-sided Video Adaptation

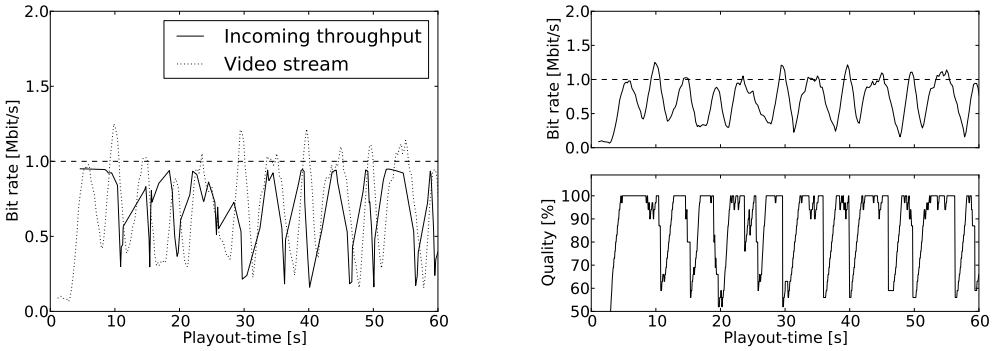
In this scenario, only the receiver scales the video stream. The receiver-sided congestion control compares the incoming rate with the current bit rate of the video stream (cf., Figure 5.11(a)) to detect an underused path. The incoming throughput varies around 1 $\frac{\text{Mbit}}{\text{s}}$. As soon as the bit rate of the video stream is lower than the incoming throughput, the receiver signals an upscaling.

The inter-arrival jitter is used as an indicator for a congested link. Figure 5.12 shows the inter-arrival jitter and the delayed frames clearly exceed the normal jitter variations after the 10 second mark.

Figure 5.11(b) shows the bit rate and the codec adaptation of the rescaled video stream. Both measurements exhibit similar behavior compared to the sender-sided codec adaptation approach.

The distribution of the inter-arrival jitter at the receiver-side is shown in Figure 5.13(a) and indicates a stable video stream transmission. Most of the frames arrive in time and provide a fluent playout of the video stream. In contrast to an unscaled video stream, the results are significantly better, but compared to the sender-sided video adaptation only 76.8 % of the frames arrive in time.

The ideal encoding quality is compared to the receiver-sided encoding suggestion and shown in Figure 5.13(b). The results of the receiver-sided approach also come close to the optimal encoding settings, which attests the efficiency of the receiver-sided approach.



(a) Incoming throughput and bit rate of the incoming video stream

(b) Bit rate and codec adaptation of the video

Figure 5.11: *Bit rate analysis of a receiver-sided video adaptation*

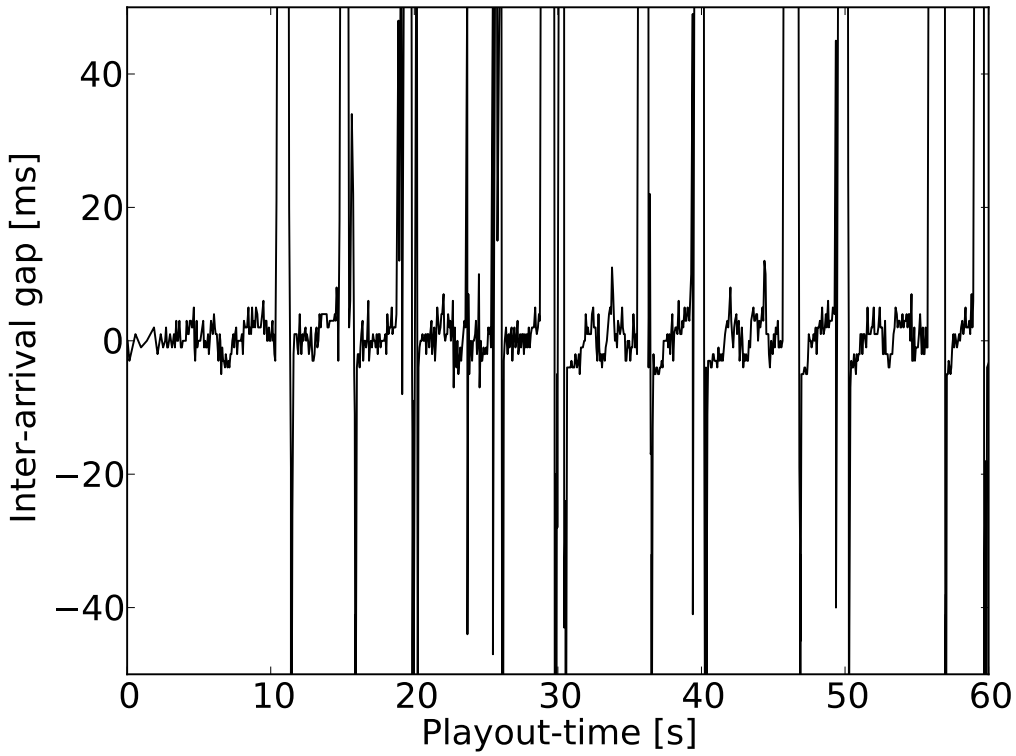
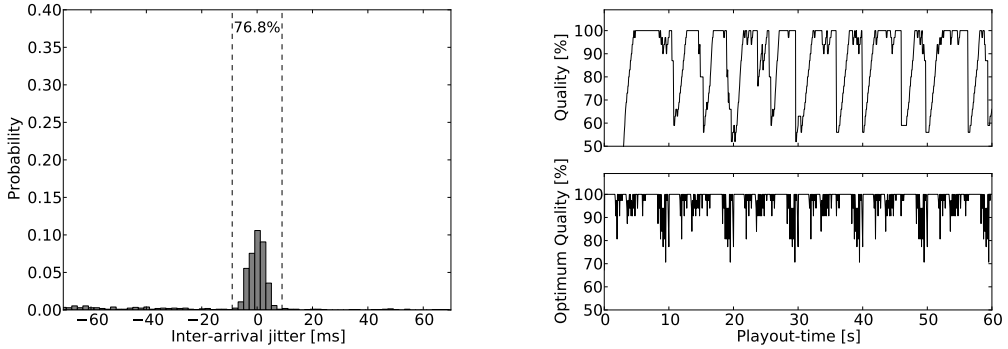


Figure 5.12: *Inter-arrival gap variation at the receiver-side*



(a) Inter-arrival jitter distribution

(b) Comparison of the encoding quality suggestion with the optimal encoding quality

Figure 5.13: *Inter-arrival jitter distribution and encoding quality analysis of a receiver-sided video adaptation*

In comparison with the sender-sided adaptation approach, the receiver-sided approach provides a more precise bandwidth estimation. Figure 5.14 shows an approximation of available bandwidth for both approaches, which is calculated with the scaling factor k and an approximated video stream bit rate. The sender-sided approach fluctuates and the estimated available bandwidth varies between 0.3 Mbit/s and 1 Mbit/s. The receiver-sided approximation shows a more precise bandwidth estimation. The estimated bandwidth is around 1 Mbit/s, unless the video stream bit rate reaches the available bandwidth and congests the path. In this case, the available bandwidth estimation drops down rapidly. The measurement results show that the sender-sided approach reacts more often, but the available bandwidth estimations are unprecise, while the receiver-sided approach reacts less often, but with a more precise bandwidth estimation.

5.5 Sender-sided and Receiver-sided Video Adaptation

In this scenario, the receiver *and* the sender influence the scaling of the video stream. Both can initiate a downscaling of the video quality (i.e., reduce the bit rate), while only the receiver is allowed to increase the video quality. If both sides suggest a downscaling in parallel or when the quality was already reduced, the lower quality value will be taken.

The sender and the receiver calculate a scaling factor k. Figure 5.15(a) visualizes the evolution of this value for both parties over time. The sender makes more downscaling requests and they are less persistent compared to the receiver-sided requests. This complies with our previous observations, which showed that the sender reacts more sensitive to network changes (cf., Section 5.3

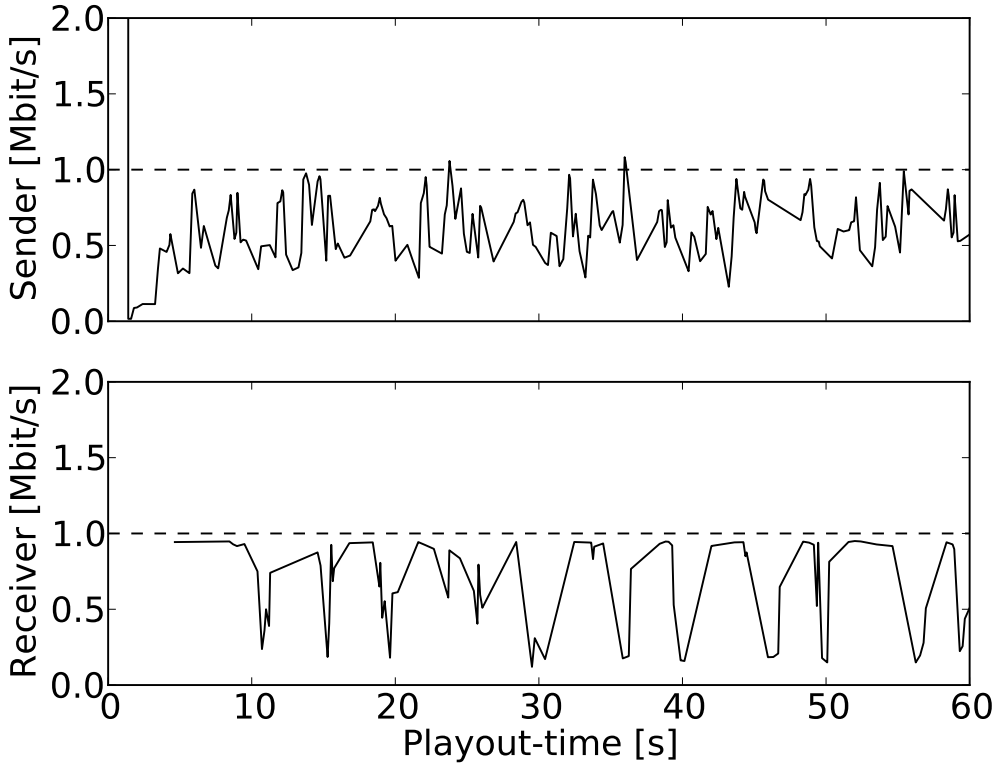
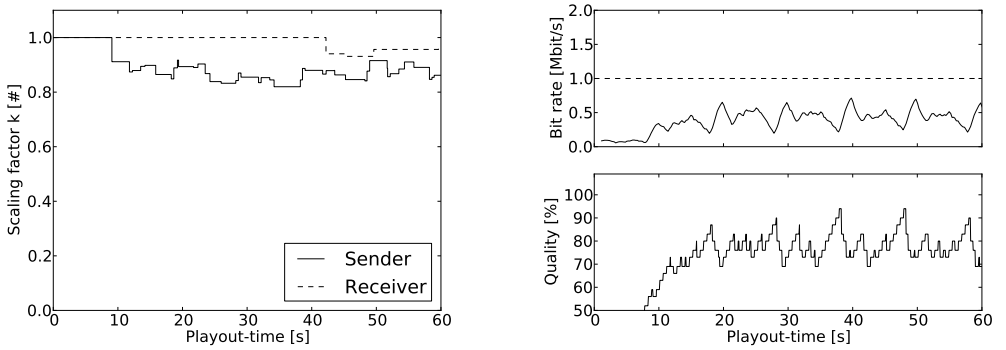


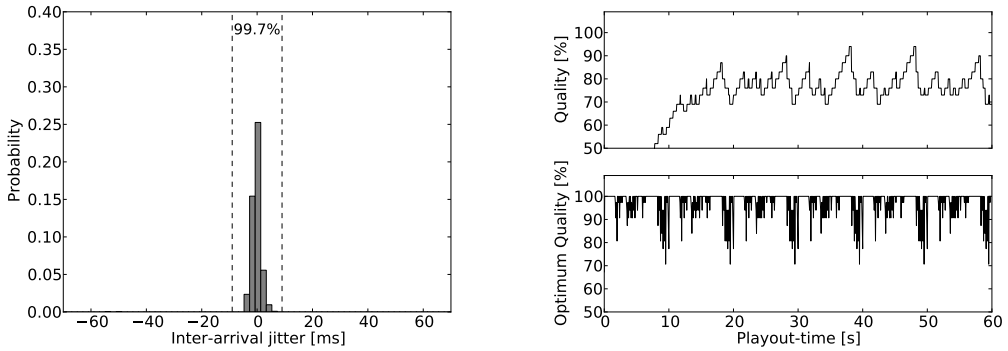
Figure 5.14: *Sender-sided available bandwidth estimation in comparison to the receiver-sided approach*



(a) Scaling suggestions for the codec adaptation

(b) Video stream bit rate and codec adaptation

Figure 5.15: *Sender-sided and receiver-sided video adaptation*



(a) Inter-arrival jitter measured at the receiver

(b) Comparison of the encoding quality suggestion with the optimal encoding quality

Figure 5.16: *Bit rate analysis for a sender-sided and receiver-sided video adaptation*

and Section 5.4). Most of the time, the sender reacts fast enough to avoid any congestion that could be detected by the receiver. There are two (rare) cases where the receiver-sided adapataion compensates the sender-sided approach. (a) The measurement period of the sender is too short and thus lacks in accuracy. (b) The sender overestimates the available bandwidth and the downscaling is not sufficient to prevent congestion. The receiver detects the underestimation and suggests an additional downscaling. The combined approach is more reliable in these situations.

Since most of the downscalings are requested by the sender, the codec adaptation looks very similar to the pure sender-based adaptation (cf., Figure 5.15(b)). Compared to the unscaled video stream, the scaled video stream exhibit still a high quality and varies between 70% and 100%. The bit rate stays below the available bandwidth and the video stream is adapted reliably. In general, the results show a fairly stable video stream transmission with just a few disturbances. Strikingly, 99.7 % of the incoming frames exhibit a jitter below 9 ms (cf., Figure 5.16(a)).

The comparison with the optimal enconding quality is shown in Figure 5.16(b). Like the sender-sided and receiver-sided approach, the characteristics of the graphs look similar. Both approaches harmonize with each other and do not inflict the encoding quality negatively.

5.6 Comparison of the Test Sequences

The quality of the sender-sided and receiver-sided approach are compared by deploying multiple video sequences in the daisy chain topology with the available bandwidth limited to 1 $\frac{\text{Mbit}}{\text{s}}$. Table

Video	Resolution	Unscaled		Scaled	
		$\bar{\beta}$ [Mbit/s]	|I.a.j.| < 9 ms	$\bar{\beta}$ [Mbit/s]	|I.a.j.| < 9 ms
TW	768x576	0.9	22.4%	0.4	99.7%
G4	768x576	1	28.4%	0.5	99.9%
KO	768x576	1.5	14.2%	0.5	99.4%
SM	768x576	0.8	66.8%	0.4	99.6%
TC	768x576	1	23.6%	0.4	99.8%
UH	768x576	4.6	0.6%	0.4	100%
Football	640x480	3.3	6.8%	1	99.8%
Washington DC	640x480	2.2	7.6%	0.8	98.3%
Akiyo	352x288	0.3	100%	0.2	99.9%
Foreman	352x288	1.4	28.2%	1	97.6%
Bus	352x288	4.1	5.4%	0.8	100%
Carphone	176x144	0.4	99.8%	0.4	99.9%

Table 5.1: *Comparison of common test sequences with an absolute inter-arrival jitter (I.a.j.) below 9 ms as rating metric*

5.1 shows average bitrate and the percentage of frames that have an absolute inter-arrival jitter (I.a.j.) below 9 ms for an unscaled and a scaled video stream.

The in due time arrival is above 95% for each video sequence. This is a huge improvement in comparison with results of the unscaled versions. The average bit rate of the video stream is lower and do not exceed the 1 Mbit⁄s. Two video sequences reach an average bit rate of 1 Mbit⁄s (Football and Foreman), which indicates that they exceed the 1 Mbit⁄s available bandwidth at some points in time. A detailed analysis showed, that the aggressive upscaling algorithm is a problem when the video stream contains a lot of movement. In these scenarios, a more reservative upscaling would be beneficial. Another video sequence with an interesting result is the video Akiyo. The unscaled video stream has an average bit rate of 0.3 Mbit⁄s and no scaling is required, but the scaled video stream only has 0.2 Mbit⁄s. This is caused by the low encoding quality at the beginning of the video transmission. After it reaches the full encoding quality no downscaling occurs.

The results show that the video adaptation works reliable independent of the video resolution or the type of the video.

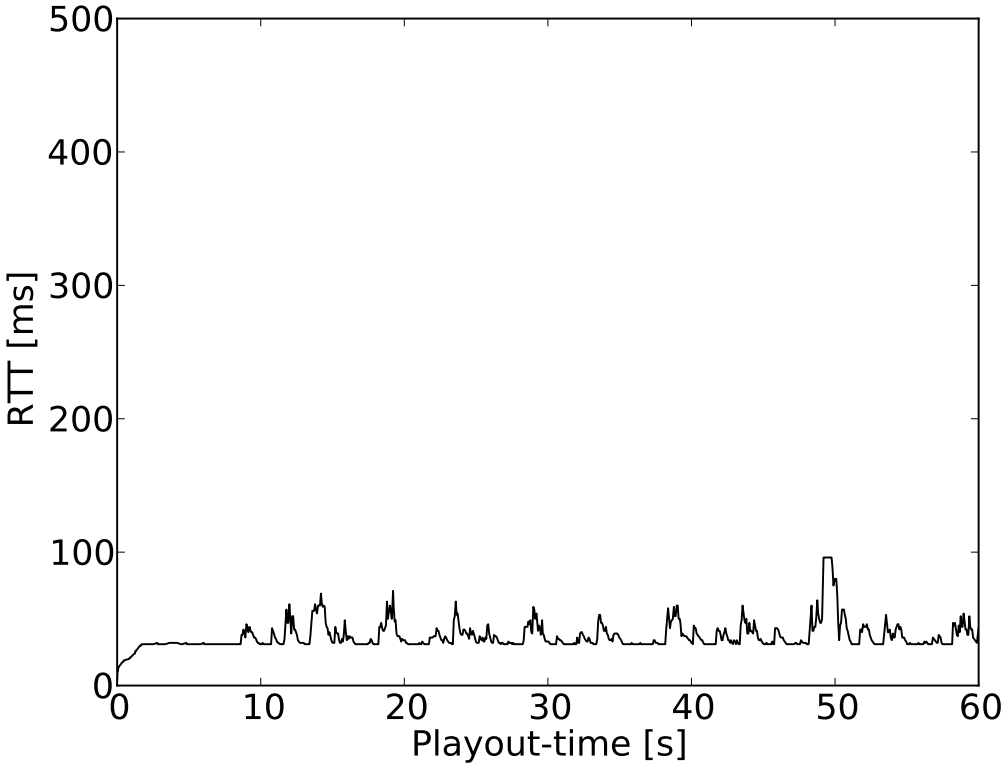


Figure 5.17: *RTT of a video stream on a path with a high loss rate*

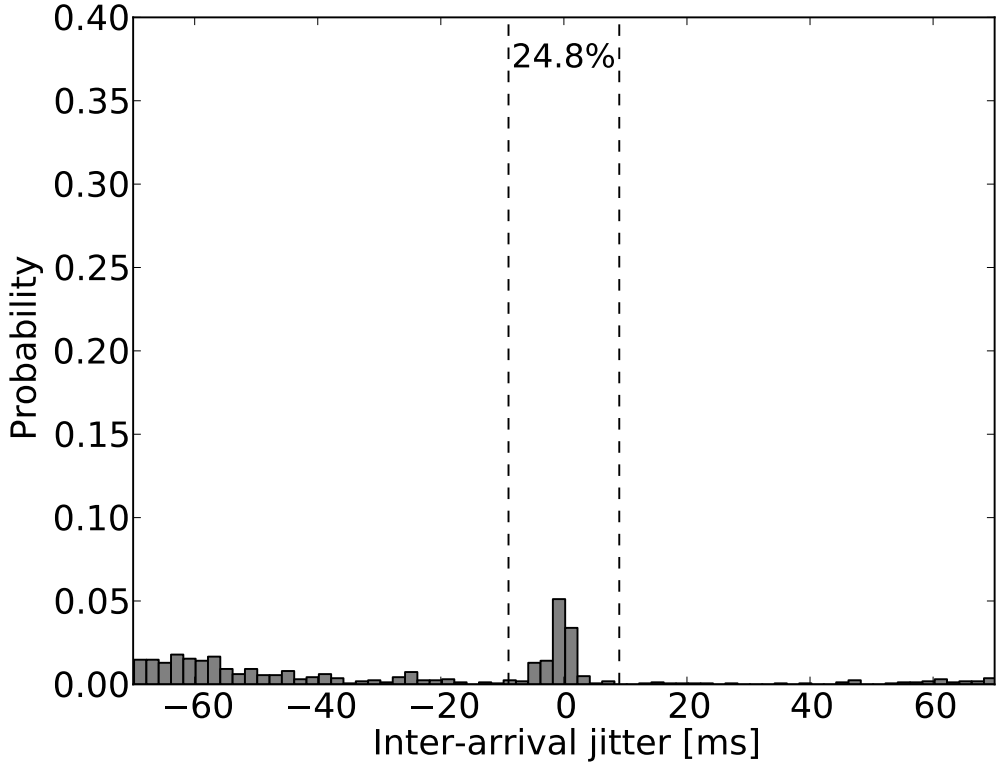
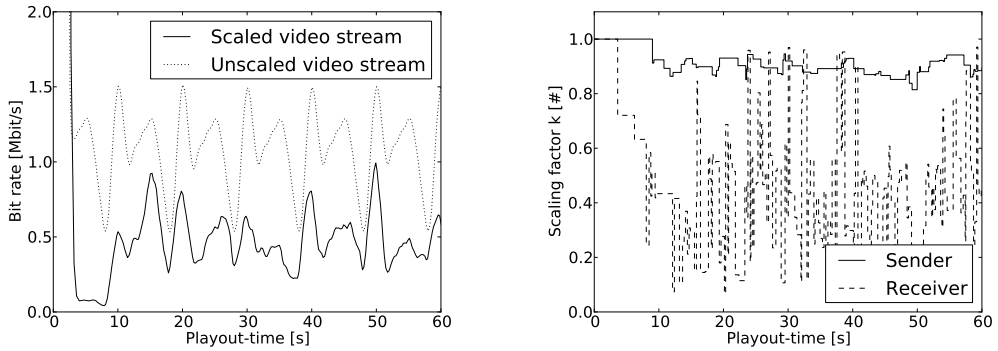


Figure 5.18: *Inter-arrival jitter distribution in a lossy environment*

5.7 Lossy paths

A further network metric that needs to be examined is the loss rate of a path. Lost packets do not influence the sender-sided video scaling approach, because the RTT is measured with ack-packets of successfully transmitted packets (cf., Figure 5.17). In this software, lost packets are retransmitted and stretch the transmission time of the frame. This influences the measurement of the inter-arrival jitter at the receiver-side. The video application is tested on the daisy-chain topology with a loss rate of 15%. This scenario emulates mobile realms, where packet loss does not necessarily imply a congested link.

The results of the packet loss and the increasing frame transmission-time are visible in the distribution of inter-arrival jitter (cf., Figure 5.18). The 15% loss rate causes a very high inter-arrival jitter at the receiver-side and just a minority of the frames arrive in due time. Every packet that needs to be retransmitted can once more experience packet loss. If the following frame after a



(a) Bitrate of the scaled video stream

(b) Scaling suggestions

Figure 5.19: *Bitrate and scaling suggestions in a lossy environment*

delayed frame arrives in due time, it usually produce a negative inter-arrival jitter (usually below -9 ms) and compensate the delay. In combination with a slight congestion, this leads to a wide distribution of transmission times.

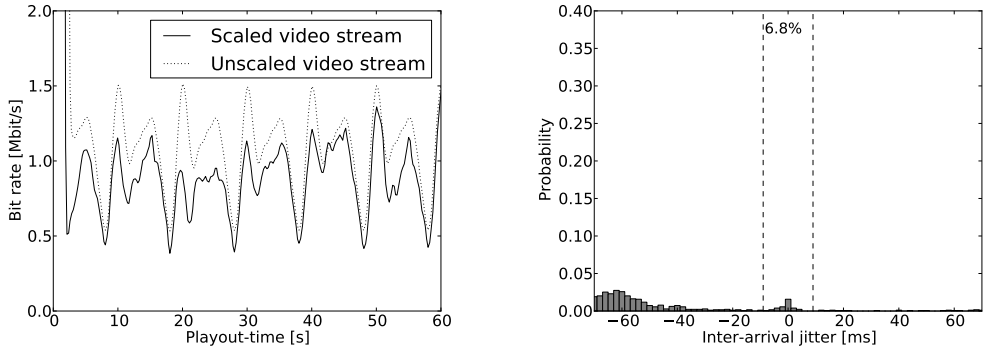
Figure 5.19(a) shows the resulting bit rate of the video stream. It shows a very low bit rate and never reaches the 1 $\frac{\text{Mbit}}{\text{s}}$ available bandwidth. In Figure 5.19(b) the scaling suggestions from both sides are shown. The receiver is responsible for the low scaling suggestions, because it detects the delayed frames and does not know about the high loss rate.

A solution to this problem is to ignore the receiver-sided scaling suggestions if the loss rate increases, but the RTT stays stable. Figure 5.20(a) shows the bit rate of the video stream if only the sender is responsible for the scaling in this scenario. This does not solve the high inter-arrival jitter variation at the receiver-side (cf., Figure 5.20(b)), but it avoids an unnecessary video stream scaling to packet loss. The high inter-arrival jitter variation can only be fixed with the retransmission timeout settings of the transport protocol.

Overall, the sender-sided approach handles packet loss much better than the receiver-sided approach. It allows a reliable scaling in a lossy environment, which is beneficial for mobile realms.

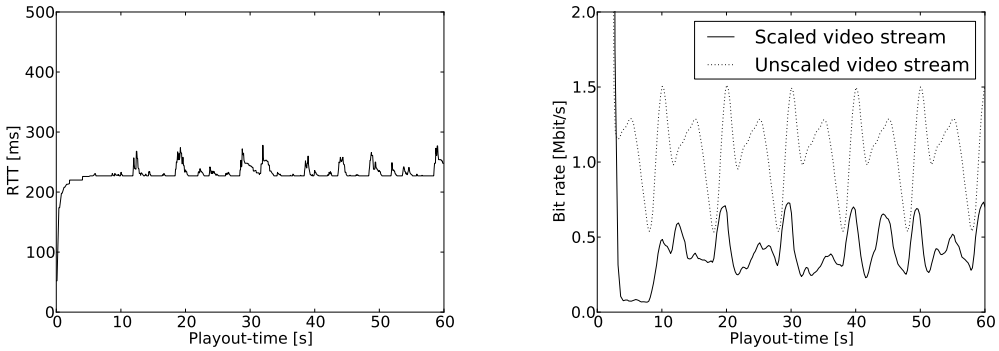
5.8 Long-RTT paths

On long distance video transmissions, the RTT is usually higher and the video adaptation is complex. The reaction time is crucial for the video adaptation to react to sudden network condition changes. Thus, the software is also tested on such a high latency path. The RTT is set to 200 ms



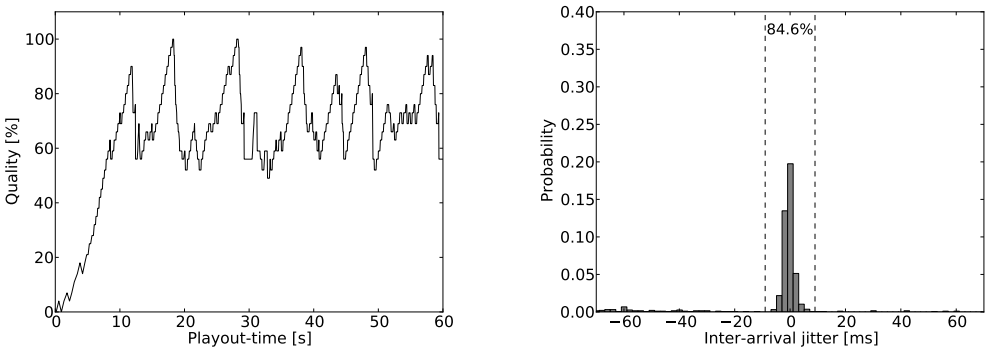
(a) Bitrate of the scaled video stream (b) Inter-arrival distribution

Figure 5.20: *Measurement results for a sender-sided scaled video stream in a lossy environment*



(a) RTT of the TW video on a path with high delay (b) Measurment results for the TW video on a path with a high RTT

Figure 5.21: *Measurement results for a network with high response times*



(a) Bitrate of the video stream

(b) Inter-arrival jitter distribution

Figure 5.22: *Measurement results for a network with high response times*

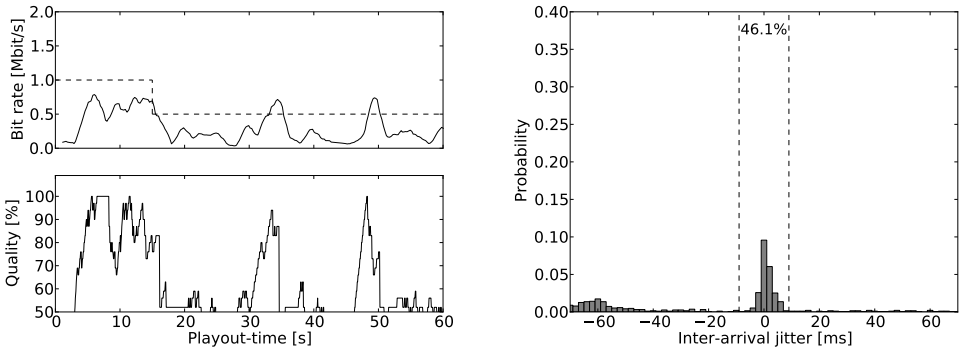
(cf. Figure 5.21(a)). In consequence, 2 further frames of a video stream with 66 ms inter-frame gap are transmitted before the first measurement results arrive (cf., equation (5.3)).

$$\frac{200ms}{66ms} > 3 \tag{5.3}$$

These two frames have the same encoding settings as the previous one. In this situation, 3 frames might be too big if the network is congested. A solution to solve this issue is to send the following frames without a higher quality setting until the measurement results arrive.

The resulting bit rate is shown in Figure 5.21(b). The low bit rate is the result of the conservative quality increasing strategy. Due to the high RTT, a high timeout is set until further quality increases are allowed (cf., Section 3.6). The quality settings for the codec in Figure 5.22(a) show the slower increasing of the quality. This behavior produces a low bit rate after a congestion is detected for a longer period of time. Unfortunately, this cannot be avoided and in these scenarios, it takes longer until a high quality is reached.

Despite the lower bit rate of the video, the QoE for the user is quite good for the given conditions. All frames arrive the receiver with a 100 ms delay, which lowers the QoE in a video conference, but is physically given by the topology. Besides that, most of the frames arrive in due time (cf. Figure 5.22(b)). Altogether, the maximum available bandwidth never exceeds 0.7 $\frac{\text{Mbit}}{\text{s}}$, but also, a congestion is successfully avoided, even with the slow response time.



(a) Bit rate and codec adaptation of the video

(b) Inter-arrival jitter distribution

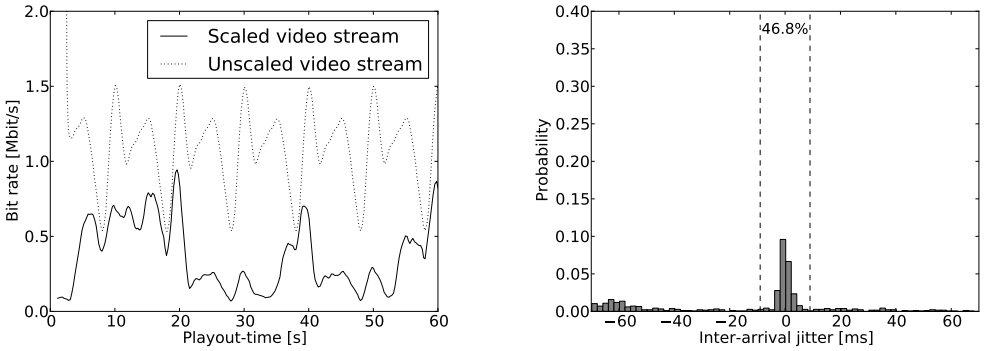
Figure 5.23: *Scaled video deployed in a dumbbell topology with 0.5 Mbit/s competing UDP traffic after 15 seconds*

5.9 Video Adaptation in the Presence of Competing Traffic

In the previous scenarios, the available bandwidth was limited by the hard-setted bottleneck. In real networks, the available bandwidth is limited by the side-traffic on a path. For the emulation of side-traffic, another test set-up is needed. The sender-sided and receiver-sided adaptation is evaluated in a dumbbell topology with competing side traffic. The bandwidth is limited to 1 Mbit/s. After 15 seconds, a 0.5 Mbit/s UDP side-traffic is initiated in parallel to the video stream. The effects of the competing side traffic are clearly visible after 15 seconds with respect to the bit rate and codec adaptation (cf., Figure 5.23(a)). The video quality is temporarily reduced to 50%. This lowers the bit rate and prevents a congestion. The inter-arrival jitter shows the influences of the side-traffic and only 46.1% of the frames arrive in due time (cf., Figure 5.23(b)).

Due to the low bit rate, every scaling step has a relative big impact on the video stream and makes it harder to archive the requested bitrate. In these conditions, a more conservative upscaling would be beneficial.

The application is also tested with 0.5 Mbit/s TCP side-traffic after the 15 second mark. In this work, we do not focus on TCP friendliness, but TCP streams are the common competing traffic streams and thus, they need to be considered. Due to the slow-start of TCP, it takes a while until the TCP affect the video stream, but after 20 seconds, the side-traffic is clearly visible in the bit rate (cf., Figure 5.24(a)). Compared to the previous UDP measurement, the bit rate is lower and the inter-arrival jitter distribution is worse (cf., Figure 5.24(b)). The congestion avoidance algorithm of the TCP protocol and the streaming application interfere with each other, which leads to worse results. The TCP stream reacts to the video stream and changes its sending rate. The resulting available bandwidth fluctuates and it is complex to scale the video stream accordingly. The quality



(a) Comparison of the scaled and unscaled video stream bit rate

(b) Inter-arrival jitter distribution

Figure 5.24: *Video stream in a dumbbell topology with 0.5 Mbit/s competing TCP traffic*

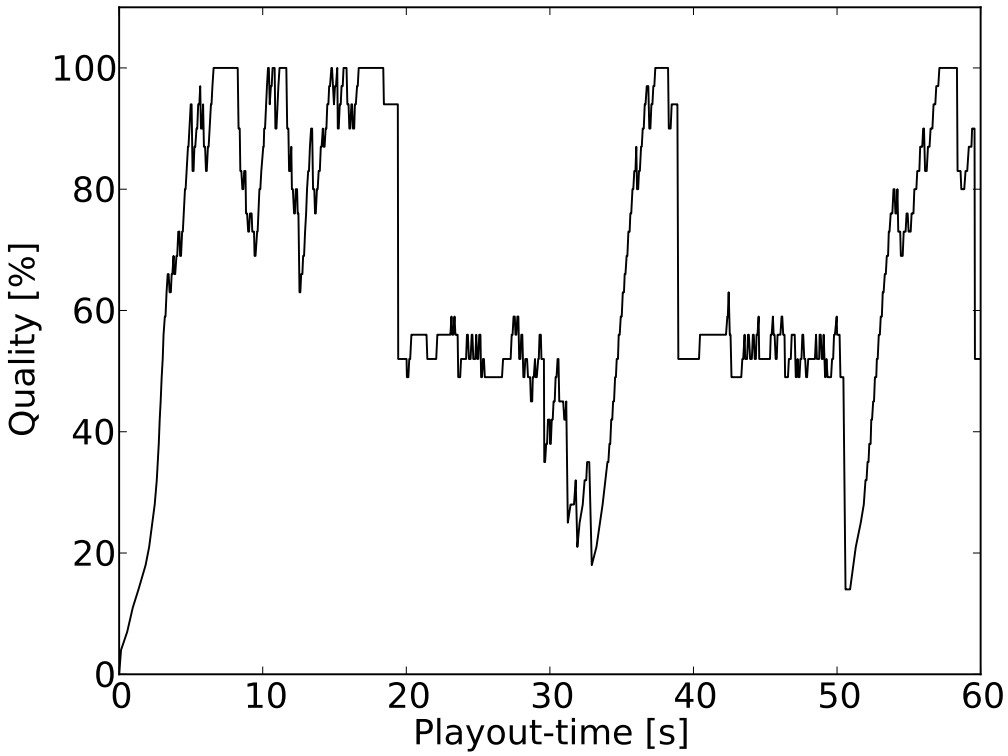
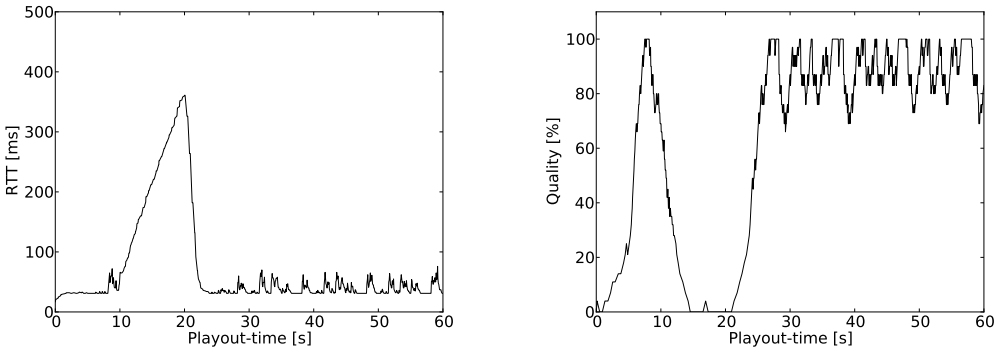


Figure 5.25: *Quality of the video stream on a path with 0.5 Mbit/s TCP side-traffic*



(a) RTT of the TW video on a path with a short peak of side-traffic after 10 seconds

(b) Bitrate and codec adaptation of the video stream

Figure 5.26: *Fully congested path*

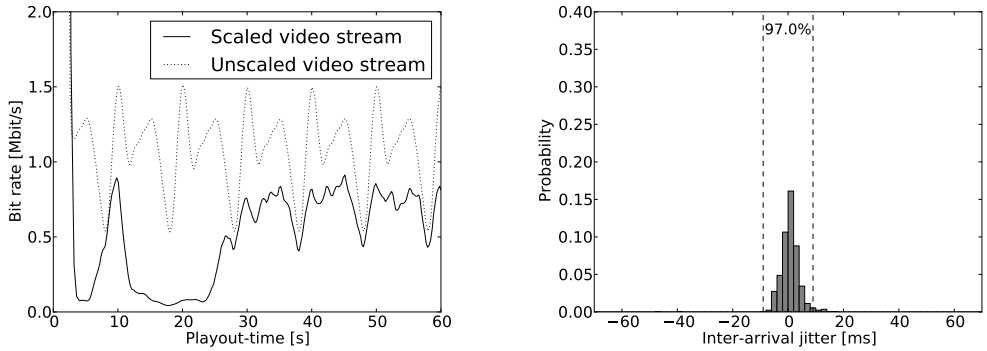
of the video stream (cf., Figure 5.25) shows more drastic downscale. In the current state, we do not focus on TCP-friendliness, but in the future the effects need to be examined and a better coexistence with TCP is aspired.

Another common side-traffic effects are side-traffic bursts. We emulate a short congestion with a short 1 Mbit/s UDP side-traffic stream at the 10 second mark. The side-traffic congests the path completely and the effects are visible in the RTT (cf., Figure 5.26(a)). The application reacts with a drastic reduction of the quality, which is shown in Figure 5.26(b). The congestion control reacts to the challenging network condtions and is also able to detect the disappearance of the congestion very fast. This allows a quick quality increasing if the path is free. The resulting bandwidth and the inter-arrival jitter distribution is show in Figure 5.27(a) and Figure 5.27(b). The congestion has a clear influence on the bit rate of the video stream, but the inter-arrival jitter does show an eligible behavior, where most of the frames arrive in due time.

5.10 Video Adaptation in a Network with a Congested Return Path

A known problem of the sender-sided video adaptation algorithm is a congested return path. A congested return path does not impair the video stream transmission, but it influences the RTT measurements. The sender-sided approach detects the RTT variations and interprets it as congestion. In this specific scenario, a downscaling of the video stream in unnecessary.

The sender-sided and receiver-sided adaptation are evaluated in a dumbbell topology with a congested return path. The bandwidth is limited to 10 Mbit/s, which is enough to send the video



(a) Bit rate of the TW video on a path with a short peak of side-traffic after 10 seconds

(b) Inter-arrival jitter distribution

Figure 5.27: *Fully congested path*

stream with the highest quality settings. The return path is limited to 1 $^{\text{Mbit}}/_{\text{s}}$. Instead of one long-lasting congestion that only cause a short sender-sided congestion detection, we sent multiple random packet bursts over the return path to produce multiple short-lasting congestion. The resulting RTT fluctuations (cf., Figure 5.28) are recognized by the sender-sided approach and it scales the video stream down.

Figure 5.29(a) shows the sender-sided scaling suggestions and Figure 5.29(b) shows the resulting quality. The receiver does not detect any congestion and no scaling suggestions are made (cf., Figure 5.29(a)). Even with the very low encoding quality, the sender-sided approach detects a congested path and constantly suggests a downscaling of the video stream. The resulting bit rate in Figure 5.30(a) is very low due to sender-sided scaling suggestions. This measurement shows an important downside of the sender-sided approach: It is unknown for the sender, if the increasing RTT is caused by the forward or the return path. Thus, a congested return path can cause a downscaling, even if it does not inflict the video stream. A measurement of the one-way delay, can avoid this behavior, but it requires synchronized clocks. The receiver-sided approach works reliable in these situations.

5.11 Video Adaptation in a Network with a Constant Increasing Congestion

The sender-sided algorithm indicates a congestion with a signficant jump in the jitter variation of the RTT. If a congestion is caused by a side-traffic stream that slightly exceeds the available bandwidth, the filling rate of the router queues is low. If the side-traffic stream has a constant

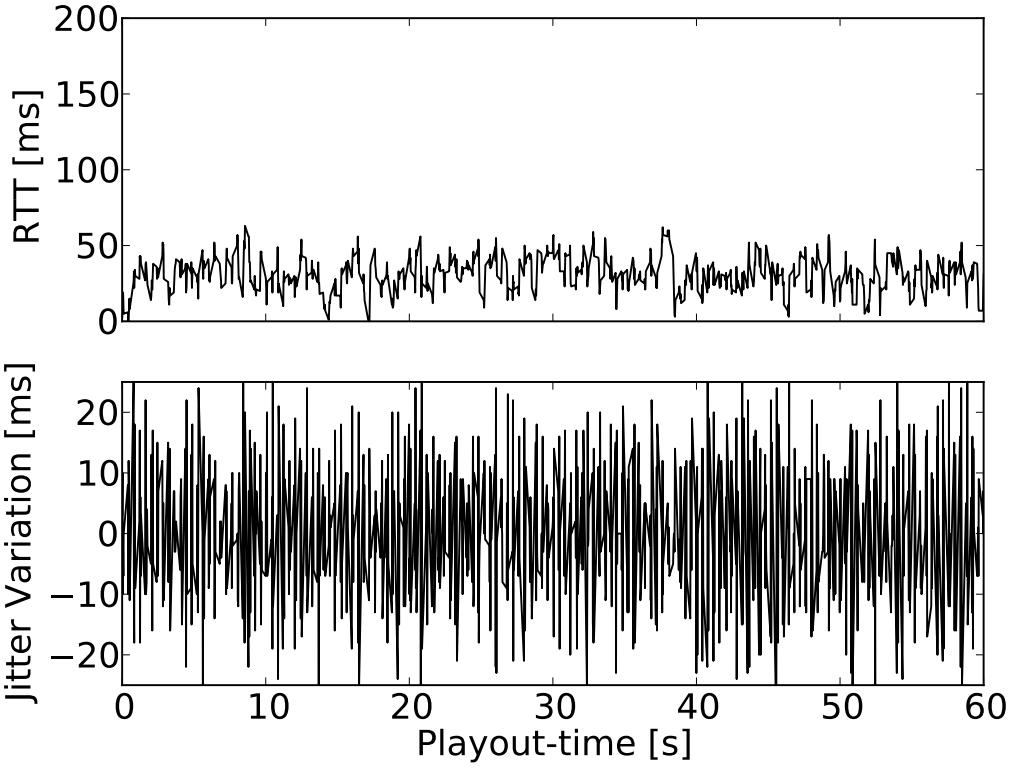
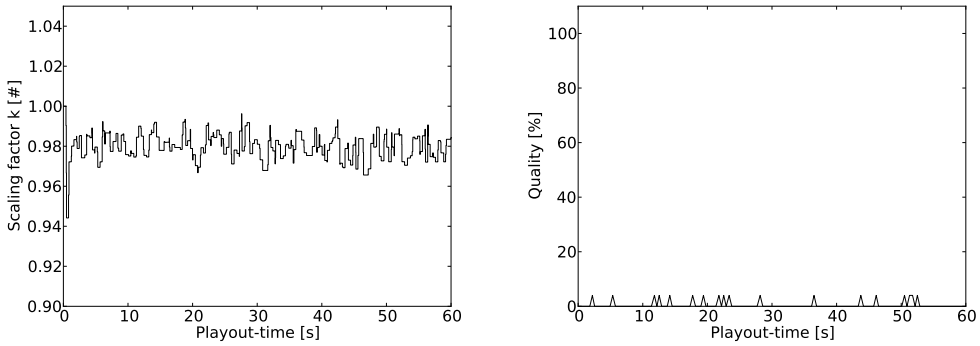
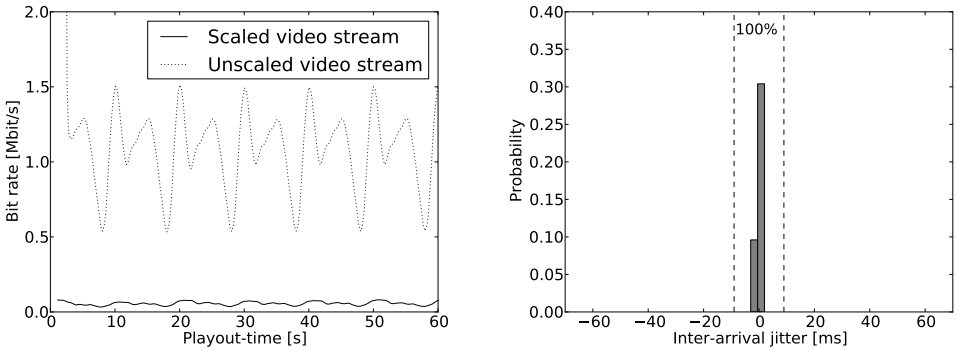


Figure 5.28: *RTT and RTT jitter variation*



(a) Scaling suggestions (b) Scaling suggestions

Figure 5.29: *RTT measurement and sender-sided scaling suggestions*



(a) Bit rate and codec adaptation of the video

(b) Inter-arrival jitter distribution

Figure 5.30: *Analysis of the video stream in a network with a congested return path*

rate, the filling rate is also constant. The RTT constantly increases and the RTT jitter variation does not show any significant jumps. As a result, the sender-sided approach has no indicator for a congested link.

The streaming application is deployed with the previously described network conditions to verify this assumption. As expected, the RTT (cf., Figure 5.31) increases with a constant rate. The jitter variation of the RTT fluctuates, but stays below the threshold of 5 ms most of the time. The resulting scaling suggestions in Figure 5.32(a) shows that the sender is not aware of the congestion and only the receiver provides reliable scaling suggestions. The resulting video stream bit rate is very low (cf., Figure 5.32(b)). In this scenario, it is impossible to prevent the congestion, because no bandwidth is available. The test results show that the sender-sided approach does not provide reliable scaling suggestions in this network conditions and the receiver-sided approach is necessary for a reliable video adaptation.

5.12 RTT Delay Variation

In wireless networks, the RTT shows usually significant variations, which influences both congestion detection approaches. On the sender-side, a congestion is identified by a significant jump in the jitter variation of the RTT. On a network with a fluctuating RTT, the sender detects many 'congestions' and may downscale the video stream. At the receiver-side, the variations of the RTT influence the arrival time of the frames. If a few packets of a frame are delayed, the whole frame does not arrive in time and the delay is visible in the inter-arrival jitter. Like the sender, the receiver will detect a 'congested' link, even if enough bandwidth is available.

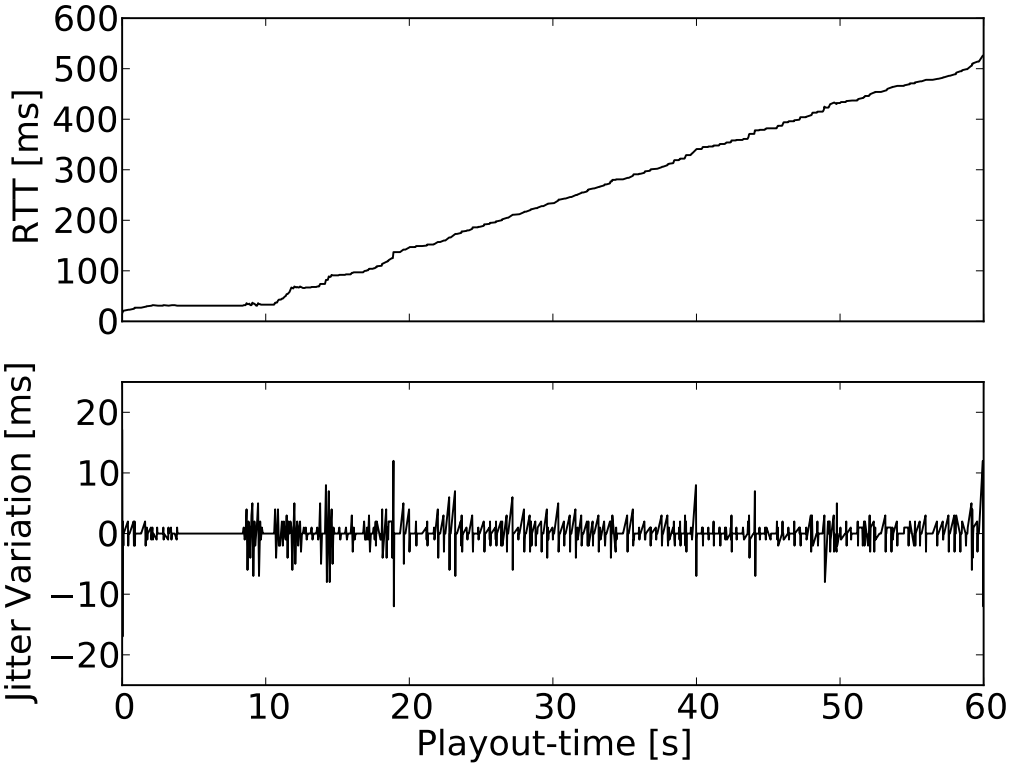
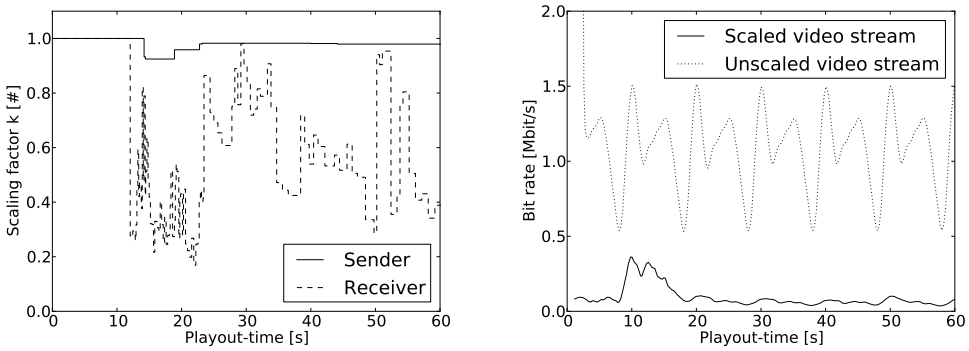


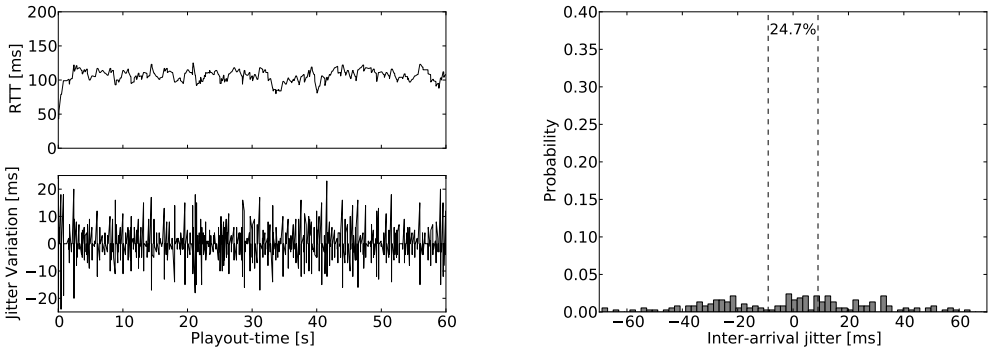
Figure 5.31: *RTT and RTT jitter variation*



(a) Scaling suggestions

(b) Bit rate of the video stream

Figure 5.32: *Analysis of the scaling suggestions*



(a) RTT and RTT jitter variation of the video stream

(b) Inter-arrival distribution

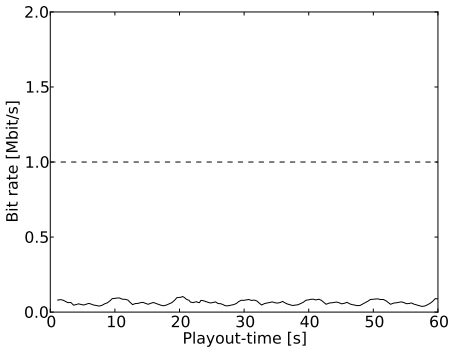
Figure 5.33: *Measurement results for a scaled video stream in a network with a volatile RTT*

The software is tested on a network with an average RTT around 100 ms that varies between 25 ms and 170 ms. The loss rate is set to 0%. These values are close to realistic values, which are experimentally determined with a mobile network (UMTS). The settings are set with the Linux traffic shaping program "tc", which allows the generation of correlating delay and pure random delays. With correlating delays, each delay relies to a certain percentage on the previous delay to emulate smoother RTT fluctuations. This is a realistic scenario in a wired network, where the delays usually depend on router queuing variations. In wireless networks, the RTT shows a more random fluctuation and we choose random delay variations to emulate this behavior. The RTT of the network is measured over 100 packets and shown in Listing 5.1.

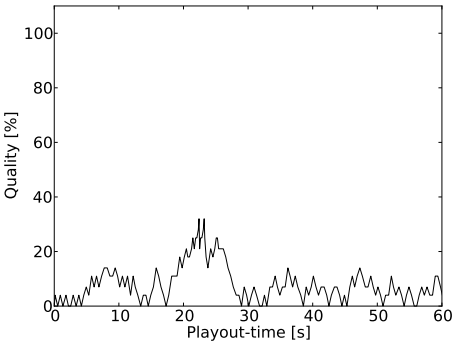
Listing 5.1: *Ping result for the test network*

```
100 packets transmitted, 100 received, 0\% packet loss, time 99120ms
rtt min/avg/max/mdev = 29.005/100.674/166.379/32.669 ms
```

A scaled video stream is sent over the network. The RTT fluctuates in this scenario and thus, the RTT jitter variation is volatile (cf., Figure 5.33(a)). The inter-arrival jitter shows a wide distribution (cf., Figure 5.33(b)). Only 24.7% of the frames arrive in due time, but the scaling algorithm cannot prevent this behavior since it is a given limitation of the network. The bit rate and the quality setting for the codecs are shown in Figure 5.34(a) and 5.34(b); Both are very low. The scaling suggestions and the resulting encoding settings produce a very low bit rate and the path is heavily underused. An analysis of the scaling factor (cf., Figure 5.35(a)) shows that the receiver-side is responsible for the low quality. The inter-arrival jitter in Figure 5.35(b), which is used as indicator for a congested link at the receiver-side, confirms this result. The inter-arrival jitter shows a wide distribution, which is caused by the delay variations for incoming frames. The sender-sided congestion control has less problems with the fluctuating RTT, even though, it uses it as indicator

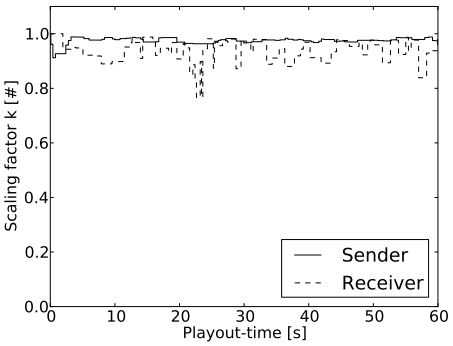


(a) Bit rate of the video stream

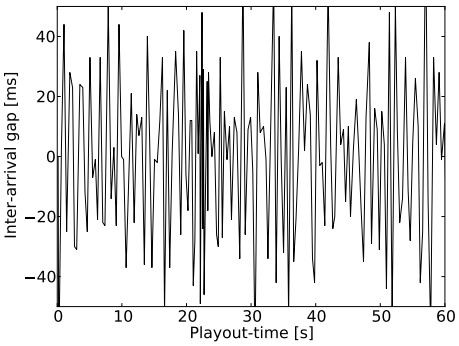


(b) Encoding quality

Figure 5.34: *Analysis of the encoding quality*

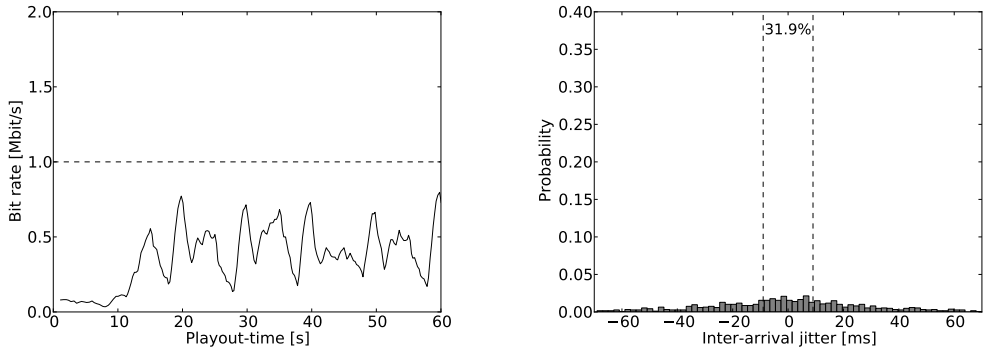


(a) Scaling suggestion from both adaptation algorithms



(b) Inter-arrival jitter

Figure 5.35: *Analysis of the scaling suggestions*



(a) Bit rate of the scaled video stream

(b) Inter-arrival distribution

Figure 5.36: *Sender-sided scaling*

for a congested link. In comparison to the receiver-sided approach, it measures the further trend of the RTT after a congestion is detected. Since the RTT does not show an increasing trend, the downscaling suggestions are rather small. To verify the theory, we only used the sender-sided approach to scale the video stream. The bit rate is shown in Figure 5.36(a) and the inter-arrival jitter distribution is shown in Figure 5.36(b). The results show a very robust scaling of the sender-sided approach, even with a fluctuating RTT. Due to the smoother scaling, the frame sizes varies less and the inter-arrival jitter is better (cf., Figure 5.36(b)). As a result, the transmission time of the video frames are more stable and the inter-arrival jitter distribution improves from 24% in due time arriving frames (cf., Figure 5.33(b)) to 31% (cf., Figure 5.36(b)).

The receiver-sided congestion control reacts too sensitive to spikes in the inter-arrival jitter. A solution to improve the receiver-sided approach in these scenarios is the usage of an average instead of a direct scaling to the inter-arrival jitter. This would make the reaction time slower, but more reliable and prevent such drastic downscaling suggestions.

5.13 Long Time Test

The previous test scenarios uses short video sequences and thus, the measurement time was very short. It allows a more clear analysis of certain effects. For a deployment in a real world application, the presented approach must work reliable over a long period of time. Thus, the software is with longer video sequences. 15 minutes of the filmlet Elephants Dream[5] is used in this measurement and the available bandwidth is set to 1.5 $\frac{\text{Mbit}}{\text{s}}$. Without scaling, the path congests

[5] http://www.elephantsdream.org

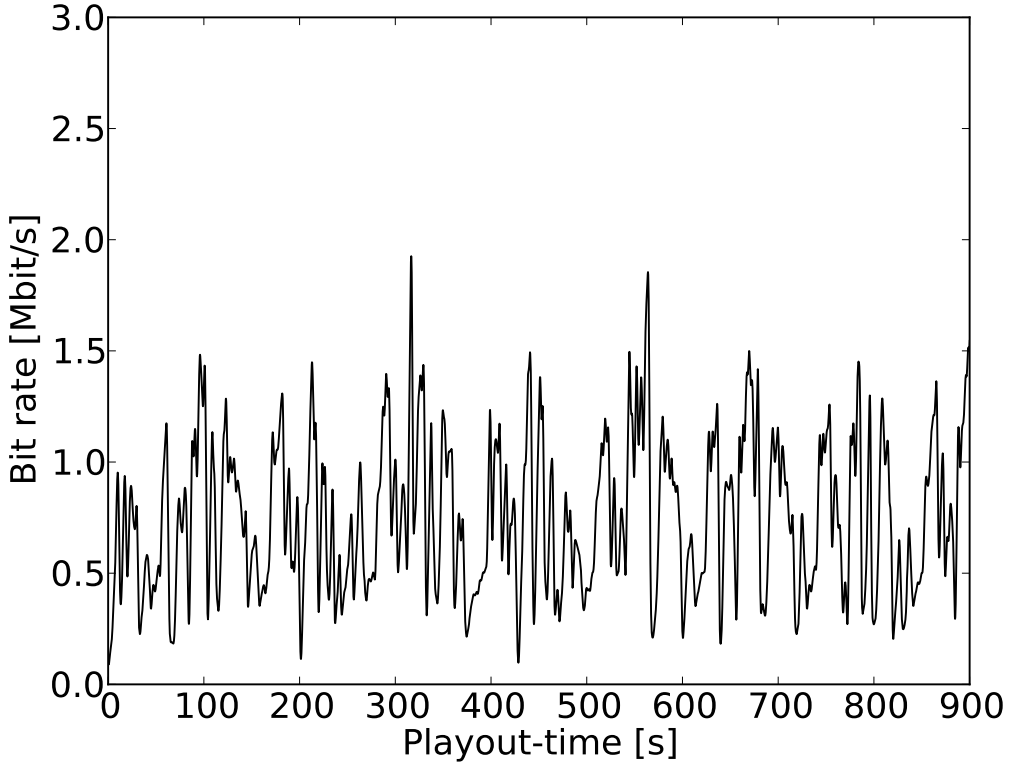


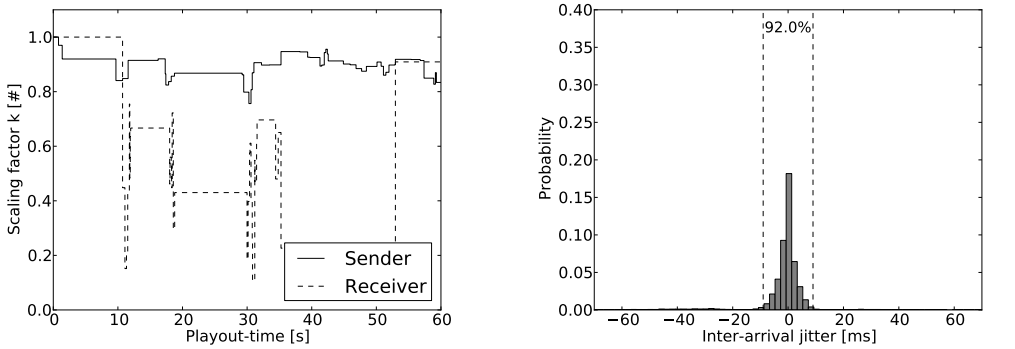
Figure 5.37: *Bit rate of the video stream 'Elephants Dream'*

within the first minute and never recovers. After a few minutes the connection disconnects. Thus, there is no point to discuss the measurement results for an unscaled video stream.

Figure 5.37 shows the bit rate of the video stream. At some points, the scaled video stream exceeds the available bandwidth (at 300s and at the 550s mark), but overall, the scaling is reliable and the provided video stream has a high quality considering the network conditions. This is a also confirmed by the inter-arrival gap of the frames (cf., Figure 5.38(b)). Almost all frames arrive in time and the user perceive only minor impairments.

5.14 Measurement in a Real Network

We have shown in the previous chapters that the scaling and adaptation approaches operate reliable in small test networks under controlled conditions. We also deployed software in a real



(a) Bitrate of the video stream 'Elephants Dream'

(b) QoE of the video stream 'Elephants Dream'

Figure 5.38: *Test results for the testsequence 'Elephants Dream' over 15 minutes*

network to evalute the approaches. Since we focus on the mobile realm, we deployed the application on a network that contains an UMTS[6] connection. The upload link is used to send the video stream, because it has a lower availbable bandwidth and the video stream needs to be scaled. The maximum upload rate is 5.76 Mbit/s with HSUPA, but the measured effective throughput varies between 1 Mbit/s and 1.7 Mbit/s.

The RTT of the path is shown in Listing 5.2 and varies around an average of 97 ms.

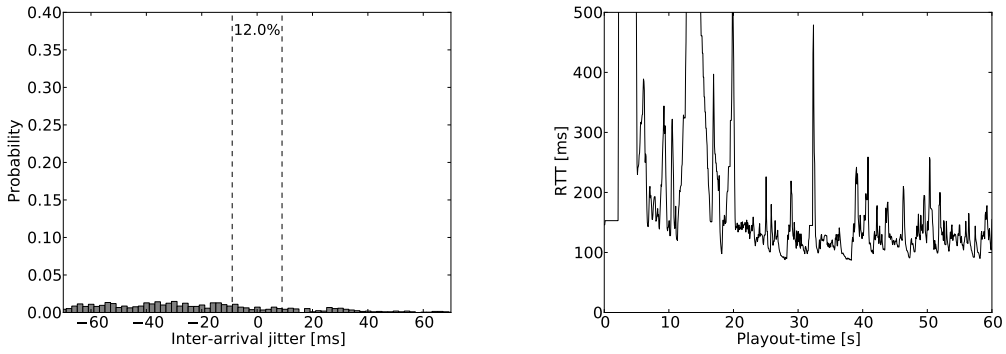
Listing 5.2: *RTT measurement over 300 seconds*

```
300 packets transmitted, 300 received, 0\% packet loss, time 299152ms
rtt min/avg/max/mdev = 60.076/96.885/1022.387/59.121 ms, pipe 2
```

The minimum RTT is 60 ms and the maximum peak is above 1 s. The average jitter of the RTT is 59.121 ms, which indicates a fluctuating RTT even when the path is free. In these network conditions, a video adaptation is difficult and the 9 ms inter-arrival jitter as quality rating is a challenging requirement to fullfill.

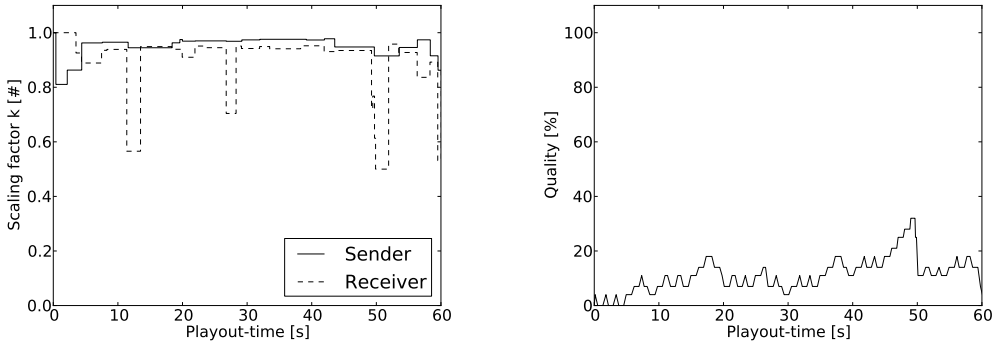
The video sequence "TW" is used for measurement. The inter-arrival jitter distribution of the incoming unscaled video stream shows huge variations (cf., Figure 5.39(a)) and only 12 % of the frames arrive in due time. The RTT shows a volatile behavior (cf., Figure 5.39(b)). The RTT variations are challenging for the sender-sided approach, because they can be interpreted as congestion. We also expected the receiver-sided approach to suggest very drastic downscaling suggestions based on the test results in Section 5.12. The measurement results confirm this assumption. Figure 5.40(a) shows the scaling suggestions. The resulting encoding quality is shown in Figure 5.40(b). Even on very low encoding settings, the receiver suggests downscaling

[6]Model: XS Stick W14



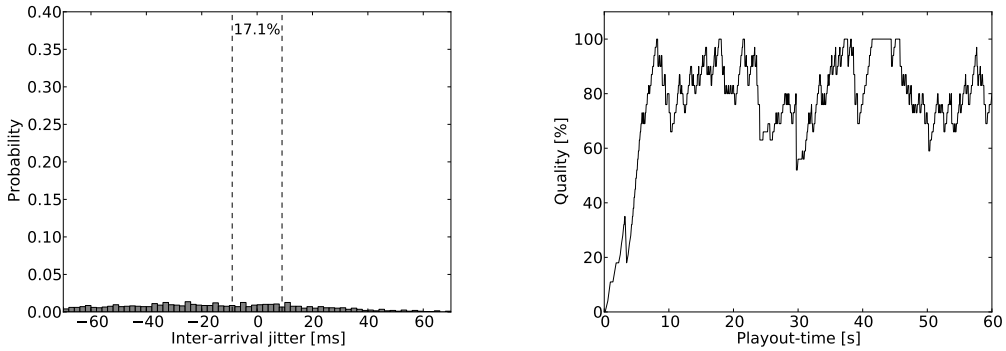
(a) Inter-arrival jitter distribution (b) RTT

Figure 5.39: *Unscaled video stream via UMTS*



(a) Scaling suggestions (b) Encoding quality of the video stream

Figure 5.40: *Sender-sided and receiver-sided scaling*



(a) Inter-arrival jitter distribution

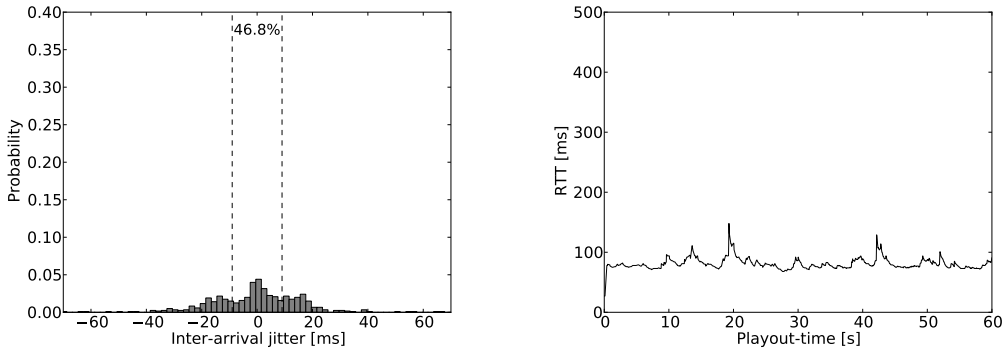
(b) Encoding quality of the video stream

Figure 5.41: *Sender-sided scaled video stream via UMTS*

factors around 0.5. The application is never able to reach a high quality stream. The resulting bit rate of the video stream never exceeds the available bandwidth, but the receiver-sided video adaptation also heavily underuses the link. The network is tested with different adaptation parameters, but the receiver-sided scaling reacts too drastic to spikes in the inter-arrival variations. For further test, only the sender-sided scaling approach is used and only the upscaling suggestions are considered from the receiver-side.

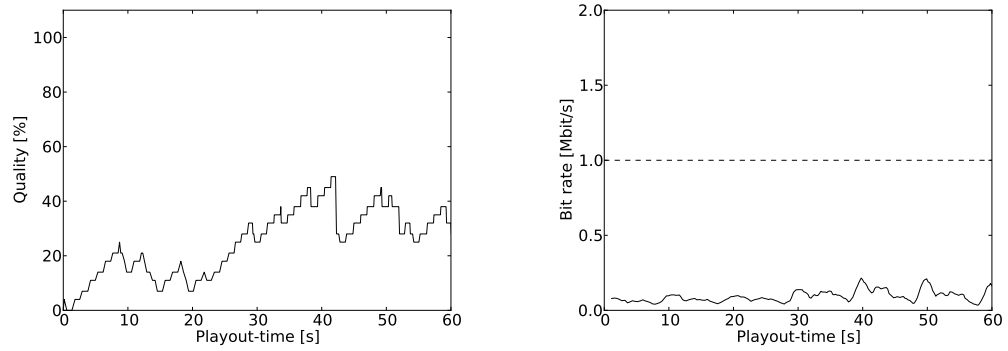
If only the sender-sided congestion control is used, the distribution of the inter-arrival jitter is slightly better (cf., Figure 5.41(a)), but still not the desired result. The problem is the aggressive upscaling (cf., Figure 5.41(b)), which cannot be countersteered by the congestion control. In these scenarios, the application needs to be parametrized with a more conservative upscaling. We set the upscaling timeout to 1 second, which means that after every upscaling, at least 1 second passes until the video stream can be upscaled again. The inter-arrival jitter distribution shows a huge improvement (cf., Figure 5.42(a)) and 48.6% of the frames arrive in due time. Also, the RTT, shown in Figure 5.42(b), is stable. The downside of this adaptation setting is visible in the quality (cf., Figure 5.43(a)) and the resulting bit rate (cf., Figure 5.43(b)). The good inter-arrival jitter distribution are the result of an underused path. The quality varies between 30% and 40% and never produces a high bit rate.

The low bit rate of the video stream is caused by the low RTT jitter variation thresholds for the sender-sided congestion detection. The sender-side identifies a congestion, when the RTT jitter variation exceeds 5 ms. It is used as indicator for a possible congestion and the quality is not always downscaled, but it sets an upscaling timeout that prevent an upscaling. The application never considers the path to be free and avoids upscaling most of the time. Several parameter combination are tested and a good tradeoff between a high bit rate and a good inter-arrival jitter distribution are 20 ms as RTT jitter variation to identify a congested path.



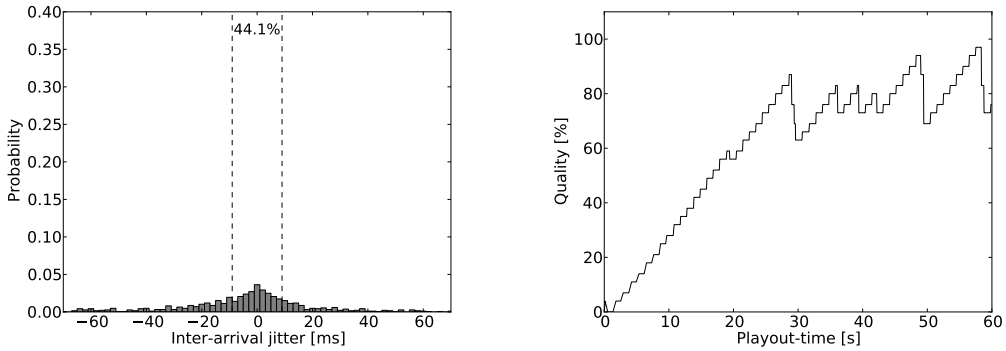
(a) Inter-arrival jitter distribution (b) RTT

Figure 5.42: *Inter-arrival jitter distribution and RTT of a sender-sided scaled video stream with a 1s upscaling timeout*



(a) Encoding quality of the video stream (b) Bitrate of the video stream

Figure 5.43: *Encoding quality and resulting bitrate of a sender-sided scaled video stream with a 1s upscaling timeout*



(a) Inter-arrival jitter distribution

(b) Encoding quality of the video stream

Figure 5.44: *Sender-sided video adaptation with a 1s upscaling timeout and 20ms threshold for the congestion indication*

The inter-arrival jitter distribution shows with 44.1% a good result (cf., Figure 5.44(a)). Keep in mind that the variations of the RTT also influences the inter-arrival jitter at the receiver-side and a very low bitrate video stream only archives 46.8% frames below 9 ms inter-arrival jitter (cf., Figure 5.42(a)). Compared to the unscaled video stream with 12% in due time arriving frames (cf., Figure 5.39(a)), it is a significant improvement. The quality settings vary between 70% and 100% (cf., Figure 5.44(b)). The resulting bit rate is shown in Figure 5.45. The measurement results show that the presented video stream adaptation approaches provide a video stream with a high QoE, if it is well parametrized.

A longer sequence of the video sequence "Elephants Dream" is used to test the adaptation approach over a longer period of time in a real network. The receiver of the video stream is a mobile device. To emulate a real world scenario, the device is not stationary, but changes its location. The movement changes the QoS of the UMTS connection. The previous experimentally determined parameters are used in this scenario (20 ms as threshold for the RTT jitter variation and a 1 s upscaling timeout). Also, the receiver-sided approach is enabled, but with a 200 ms inter-arrival jitter threshold to detect a congestion. We expect a low video stream bit rate due to radical downscaling suggestions from the receiver-side when some frames arrive experience extraordinary delays. The bit rate in Figure 5.46(a) shows a stable video stream that varies between 0.3 Mbit/s and 0.5 Mbit/s. The distribution of the inter-arrival jitter in Figure 5.46(b) is smooth and 38.2% of the frames arrive in due time. An analysis of the encoding quality (cf., Figure 5.47(a)) shows that the sender-sided approach is responsible for most of the scaling decision, while the receiver-sided provides less, but more radical suggestions (cf., Figure 5.47(b)). The results show that the video adaptation ensures a reliable video stream adaptation without congesting the path.

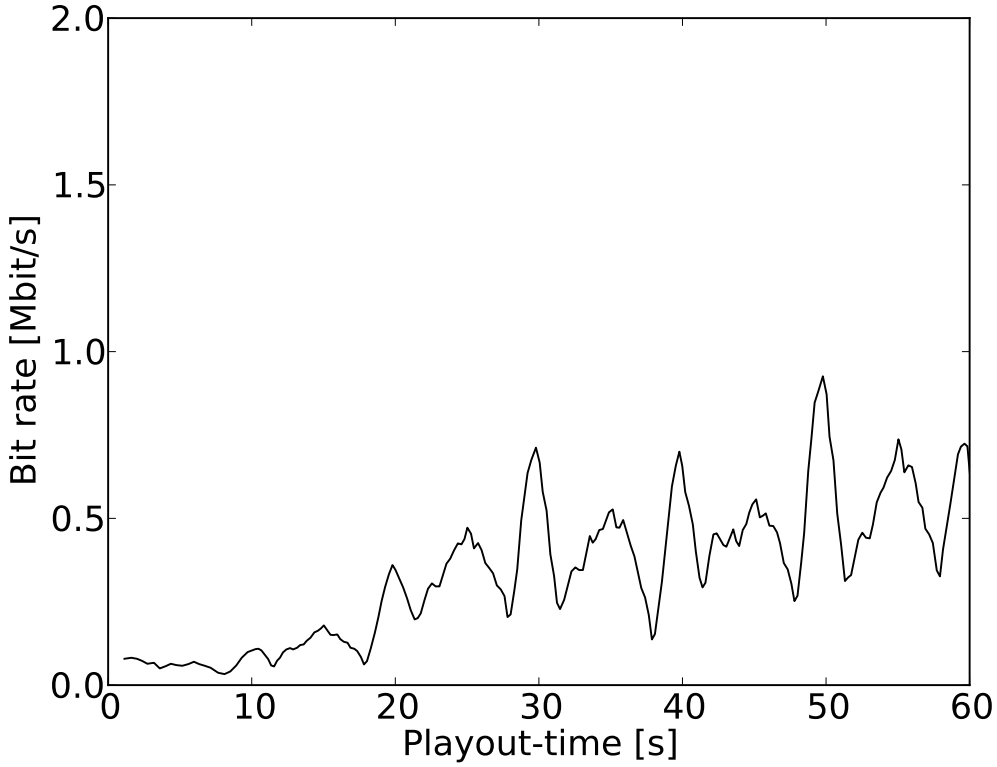
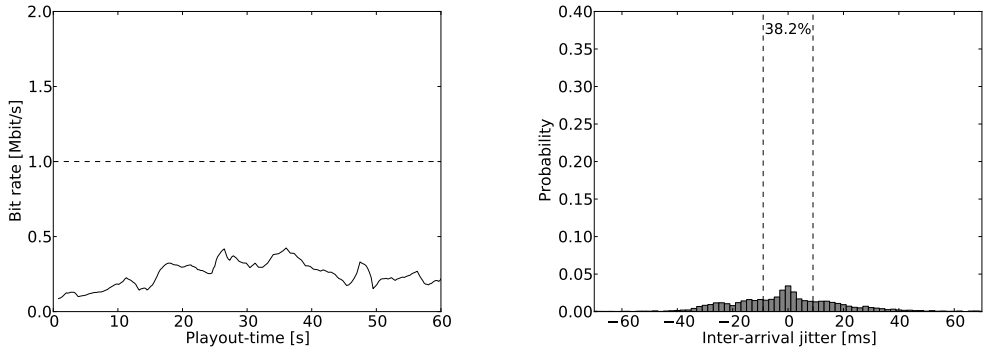


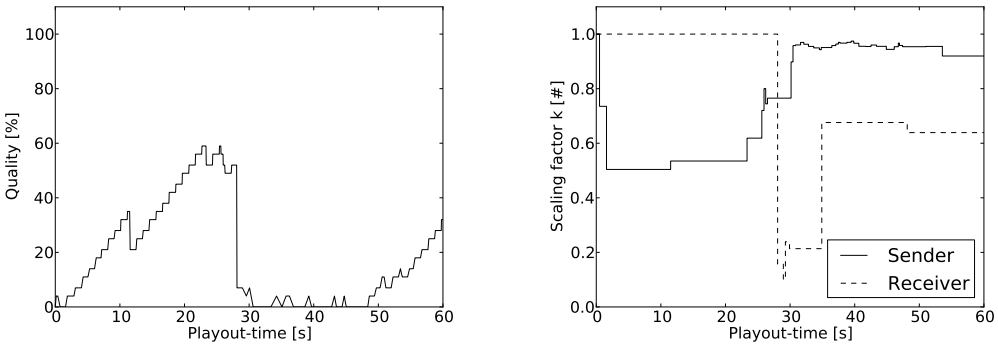
Figure 5.45: *Bit rate of the sender-sided scaled video stream with 1s upscaling timeout and 20ms RTT jitter variation threshold for the sender-sided congestion indication*



(a) Bit rate of the video stream

(b) Inter-arrival jitter distribution

Figure 5.46: *Video sequence 'Elephants Dream' in a real world deployment*



(a) Encoding quality of the video stream

(b) Scaling suggestions

Figure 5.47: *Encoding quality analysis of the video sequence 'Elephants Dream' in a real world deployment*

5.15 Analysis and Comparison of the Measurement Results

The video adaptation approaches are tested in different controlled topologies with different network settings. Additionally, the application is tested in a real world deployment. This wide variety of different network condtions is used to examine the robustness and reliability of the scaling approaches in networks with different QoS impairments. Both approaches are analyzed seperately and show different problems.

The sender-sided approach uses variations in the jitter of the RTT to detect a congested path. Thus, the sender reacts very fast to changing network conditions and provide a suitable approach for real-time multimedia applications. The sender-sided approach shows reliable measurement results as long as the return path is not congested (cf., Section 5.10) or the RTT shows a slow and constant increasing without a significant jump in the jitter variation (cf., Section 5.11). A congested return path interfere the RTT measurement, even if the video stream is not disturbed and causes unnecessary downscaling. A slowly congesting path causes only slight variations in the RTT jitter and the sender-sided approach is not able to detect the congestion. In this case, the path congests without any reactions from the sender-side. Besides that, the sender-sided approach shows reliable results, even in networks with packet loss or in presence of side-traffic.

The receiver-sided approach uses significant inter-arrival jitter variations to detect a congested path. It uses whole frames instead of packets for the measurement, which provides a slower, but more accurate congestion detection (cf., Section 5.4). The measurement results also show that the receiver-sided approach reacts too sensitive to RTT variation (cf., Section 5.12) and suggests a too drastic downscaling of the video stream. If one packet of a frame experience packet loss or a

huge delay, the whole frame is delayed and causes a drastic downscaling suggestion. The usage of a mean inter-arrival jitter could improve the scaling suggestions by avoiding too radical down scaling suggestions. Additionally, the incoming throughput is measured to identify an underused path, which allows a safe upscaling. In comparison with the sender-sided approach, the receiver-sided congestion control operates reliable on a congested path and is resistant to a congested return path.

Both approaches complement each other and the coexistence does not lower their effectiveness. A combination of both adaptation approaches ensures a reliable video stream adaptation in a real world network with a low QoS (compared to wired networks). The analysis of the measurements show that a coexistence of both approaches improve the performance in difficult network conditions, while neglecting their disadvantages. The sender-sided approach reacts faster than the receiver-sided approach, who only reacts if the sender is not able to prevent a congestion, but the receiver provides safe upscaling suggestions, which is not provided by the sender-sided approach.

The measurement results also show that the video application needs to be well parametrized to provide a reliable video adaptation. The experimentally determined parameters are used in a real world deployment that contains a UMTS link as limiting factor for the bit rate of the video stream. The resulting video adaptation provide a fluent video stream over a long period of time without congesting the path.

6 Conclusion and Outlook

Multimedia applications with high quality video streams are likely to cause network congestions and thus need to be aware of the network conditions to sustain fluency. Particularly on access links or in the presence of heavily changing traffic, a quick reaction is crucial for real-time multimedia applications more beneficial than a slow but more accurate video adaptation. In this work, a combination of a receiver-sided and a sender-sided congestion detection approach combined with scalable video adaptation is presented to address this problem.

Our algorithms are simple, non-intrusive and easy to implement. For a proof-of-concept and for detailed experimental evaluations, we developed a scalable conferencing application based on the DSVC codec and previous work. The application encodes a raw video, sends it over the network and detects impending congestions at both ends. Our test results give evidence that our approaches ensure a fast and reliable video stream adaptation and are capable of detecting a link congestion early enough to avoid it so that it remains unnoticeable to users. In particular, our sender-sided approach is capable of reacting to changing network conditions very fast, while the receiver-sided approach is more precise and is able to detect an underused link. In a real network, the measurement results shows that a careful parametrizing can improve the video adaptation and provide optimal results even in challenging network conditions.

The measurement results show that the adaptation approaches leaves room for improvement. The receiver-sided approach overreacts to extraordinary delayed frames, which results in extreme downscaling suggestions. A filtered measurement, which detects and dismiss extraordinary delayed frames would provide a more stable video bit rate. The sender-sided video adaptation does not work on a congested path. An additional observation of the loss rate and the sending queue could solve this issue. Another area that can be improved is the upscaling of the video bit rate. In the current state, only the receiver-sided measurements are used to decide if an unscaling is safe. To improve the upscaling, the sending queue and the loss rate could also be considered for a safe upscaling.

In the future, the software will be tested on various real networks to estimate the effects of different network conditions. Analyzing various video samples and different network conditions will help to improve the software and ensure reliable video adaptation in different environments. Important factors in this regard are the different types of side-traffic that influence the video stream. In the Internet, various applications with different traffic characteristics and protocols compete with each other. The presented video streaming application also influences competing traffic streams

and could harm other applications. These effects needs to be analyzed. Especially, the effects in a co-existence with TCP streams is important since it is the most common protocol in the Internet. TCP also uses a congestion avoidance mechanism and will reduce its sending rate if a path impend to congest. Thus, it is possible to push TCP streams aside with a UDP based streaming application. For a co-existence with other applications and protocols, these aspects require further research.

The most common protocol to transmit multimedia data is RTP. In our video streaming software, RTP is not used and instead an own implementation on top of the enet protocol is used. The reliable packet transmission improves the QoE in a video conferencing software. However, the presented algorithm do not depend on a specific protocol and only require a response channel. Thus, it is possible to use the algorithms on top of a RTP/RTCP implementation. In the future, we will present such an application.

Adapting the video bit rate to the available bandwidth in general is a wide area of research and several approaches exists how to scale multimedia streams to network conditions. We will compare the presented approach to other adaptation approaches and analyze the results.

Bibliography

[1] Advanced Video Coding for Generic Audiovisual Services. Technical Report Recommendation H.264 & ISO/IEC 14496-10 AVC, v3, ITU-T, 2005.

[2] 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.

[3] Sanjeewa Athuraliya, S.H. Low, V.H. Li, and Qinghe Yin. Rem: active queue management. *IEEE Network*, 15(3):48–53, 2001.

[4] T. Barzuza, S. Ben Zedeff, O. Modai, L. Vainbrand, Y. Wiener, and E. Yellin. Trend: A dynamic bandwidth estimation and adaptation algorithm for real-time video calling. In *Packet Video Workshop (PV), 2010 18th International*, pages 126–133, 2010.

[5] Wu chang Feng, K.G. Shin, D.D. Kandlur, and D. Saha. The blue active queue management algorithms. *IEEE/ACM Transactions on Networking*, 10(4):513–528, 2002.

[6] Kuan-Ta Chen, Cheng-Chun Tu, and Wei-Cheng Xiao. Oneclick: A framework for measuring network quality of experience. In *IEEE INFOCOM 2009*, pages 702–710, 2009.

[7] Ni Chen, Xiuhua Jiang, and Caihong Wang. Impact of packet loss distribution on the perceived iptv video quality. In *2012 5th International Congress on Image and Signal Processing (CISP)*, pages 38–42, 2012.

[8] Ni Chen, Xiuhua Jiang, Caihong Wang, and Jia Su. Study on relationship between network video packet loss and video quality. In *2011 4th International Congress on Image and Signal Processing (CISP)*, volume 1, pages 282–286, 2011.

[9] 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.

[10] 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.

[11] Qin Dai and R. Lehnert. Prediction of video perceptual quality in the presence of packet loss. In *Proceedings of the 2011 11th International Conference on Telecommunications (ConTEL)*, pages 495–502, 2011.

[12] 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.

[13] Xunli Fan, Jie Zhang, Lin Guan, and Xingang Wang. Qblue: A new congestion control algorithm based on queuing theory. In *2012 IEEE 14th International Conference on High Performance Computing and Communication 2012 IEEE 9th International Conference on Embedded Software and Systems (HPCC-ICESS)*, pages 1253–1257, 2012.

[14] Xunli Fan, Feng Zheng, Lin Guan, Jie Wang, Li Gao, and Xingang Wang. Study of ared algorithm based on the 2nd order difference equation. In *Proceedings of the 5th International Conference on Queueing Theory and Network Applications*, QTNA '10, pages 110–117, New York, NY, USA, 2010. ACM.

[15] Ivan Fernandez, Christophe De Vleeschouwer, George Toma, and Laurent Schumacher. An Interactive Video Streaming Architecture Featuring Bitrate Adaptation. *Journal of Communications*, 7(4), April 2012.

[16] M. Fiedler, T. Hossfeld, and P. Tran-Gia. A generic quantitative relationship between quality of experience and quality of service. *IEEE Network*, 24(2):36–41, 2010.

[17] Sally Floyd and Van Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4):397–413, August 1993.

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

[19] Cheng Peng Fu and S.C. Liew. Tcp veno: Tcp enhancement for transmission over wireless access networks. *IEEE Journal on Selected Areas in Communications*, 21(2):216–228, 2003.

[20] 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.

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

[22] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: a new TCP-friendly high-speed TCP variant. *SIGOPS Oper. Syst. Rev.*, 42(5):64–74, July 2008.

[23] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 3448, IETF, January 2003.

[24] T. Hendrawan and I. Mahadhika. Video quality assessment of h.264 spatial scalable video coder. In *2012 7th International Conference on Telecommunication Systems, Services, and Applications (TSSA)*, pages 306–313, 2012.

[25] Ningning Hu and Peter Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE Journal on Selected Areas in Communications*, 21:879–894, 2003.

[26] Te-Yuan Huang, Nikhil Handigol, Brandon Heller, Nick McKeown, and Ramesh Johari. Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard. In *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, IMC '12, pages 225–238, New York, NY, USA, 2012. ACM.

[27] M. Imai, Y. Sugizaki, and K. Asatani. A new estimation method using RTT for available bandwidth of a bottleneck link. In *Information Networking (ICOIN), 2013 International Conference on*, pages 529–534, 2013.

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

[29] Mapping function for transforming P.862 raw result scores to MOS-LQO. ITU-T Recommendation P.862.1, November 2003.

[30] ITU-T Recommendation P.910. Subjective video quality assessment methods for multimedia applications. Technical report, International Telecommunication Union, Geneva, Switzerland, April 2008.

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

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

[33] 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.

[34] Euy-Doc Jang, Jae-Gon Kim, Truong Cong Thang, and Jung-Won Kang. Adaptation of Scalable Video Coding to packet loss and its performance analysis. In *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, volume 1, pages 696–700, 2010.

[35] JPEG committee. JPEG 2000. http://www.jpeg.org/jpeg2000/, 2009.

[36] S.H. Kamali, M. Hedayati, A.S. Izadi, and H.R. Hoseiny. The monitoring of the network traffic based on queuing theory and simulation in heterogeneous network environment. In *ICCTD '09. International Conference on Computer Technology and Development*, volume 1, pages 322–326, 2009.

[37] Sebastain Kaune, Matthias Wählisch, and Konstantin Pussep. Modeling the Internet Delay Space and its Application in Large Scale P2P Simulation. In Klaus Wehrle, Mesut Günes, and James Gross, editors, *Modeling and Tools for Network Simulation*, pages 427–446. Springer, Heidelberg, 2010.

[38] J. Krejci. Mdi measurement in the iptv. In *IWSSIP 2008. 15th International Conference on Systems, Signals and Image Processing*, pages 49–52, 2008.

[39] Sung-Ju Lee, Puneet Sharma, Sujata Banerjee, Sujoy Basu, and Rodrigo Fonseca. Measuring Bandwidth Between PlanetLab Nodes. In Constantinos Dovrolis, editor, *6th Intern. Workshop on Passive and Active Network Measurements (PAM'05)*, volume 3431 of *LNCS*, pages 292–305, 2005.

[40] 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, August 2007.

[41] Qixiang Pang, S.C. Liew, Cheng Peng Fu, Wei Wang, and V.O.K. Li. Performance study of tcp veno over wlan and red router. In *GLOBECOM '03 Global Telecommunications Conference*, volume 6, pages 3237–3241 vol.6, 2003.

[42] Roger Pantos. HTTP Live Streaming. Internet-Draft – work in progress 12, IETF, October 2013.

[43] V. Paxson and M. Allman. Computing TCP's Retransmission Timer. RFC 2988, IETF, November 2000.

[44] C. Perkins and T. Schierl. Rapid Synchronisation of RTP Flows. RFC 6051, IETF, November 2010.

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

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

[47] ITU-T Recommendation J. 144. Objective perceptual video quality measurement techniques for digital cable television in the presence of a full reference, March 2001.

[48] ITU-T Recommendation J. 247. Objective perceptual multimedia video quality measurement in the presence of a full reference, August 2008.

[49] ITU-T Recommendation J. 340. Reference algorithm for computing peak signal to noise ratio of a processed video sequence with compensation for constant spatial shifts, constant temporal shift, and constant luminance gain and offset , June 2010.

[50] 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.

[51] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, IETF, July 2003.

[52] 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.

[53] Patrick Seeling and Martin Reisslein. Video transport evaluation with h.264 video traces. *Communications Surveys Tutorials, IEEE*, 14(4):1142 –1165, quarter 2012.

[54] 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.

[55] G. Toma, L. Schumacher, and C. De Vleeschouwer. Offering streaming rate adaptation to common media players. In *Multimedia and Expo (ICME), 2011 IEEE International Conference on*, pages 1–7, July 2011.

[56] 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.

[57] Christian J. van den Branden Lambrecht and Olivier Verscheure. Perceptual quality measure using a spatio-temporal model of the human visual system. In *Proc. SPIE Internat. Conf. on Digital Video Compression: Algorithms and Technologies*, volume 2668, 1996.

[58] M. Venkataraman and M. Chatterjee. Inferring video qoe in real time. *Network, IEEE*, 25(1):4–13, 2011.

[59] J. Vieron and C. Guillemot. Real-time constrained TCP-compatible rate control for video over the Internet. *Multimedia, IEEE Transactions on*, 6(4):634–646, August 2004.

[60] J. Welch and J. Clark. A Proposed Media Delivery Index (MDI). RFC 4445, IETF, April 2006.

[61] H.R. Wu, T. Ferguson, and B. Qiu. Digital video quality evaluation using quantitative quality metrics. In *Fourth International Conference on Signal Processing Proceedings, 1998. ICSP '98.*, volume 2, pages 1013–1016 vol.2, 1998.

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §22(4) bzw.§24(4) ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 26.03.2014 Fabian Jäger