How Dia-Shows Turn Into Video Flows: Adapting Scalable Video Communication to Heterogeneous Network Conditions in Real-Time

Fabian Jäger, Thomas C. Schmidt
fabian.jaeger@haw-hamburg.de, t.schmidt@ieee.org
iNET Research Group – Department Informatik
Hamburg University of Applied Sciences
Berliner Tor 7, 20099 Hamburg, Germany

Matthias Wählisch
waehlisch@ieee.org
Institut für Informatik
Freie Universität Berlin
Takustr. 9, 14195 Berlin, Germany

***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 analyse 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 sustained goodput to extract a feasible scaling. In thorough evaluations that include real-world 3G networks, we reveal a faster congestion detection at the sender that are also more robust but less accurate than probing at the receiver.**

I. 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 messaging, and mobile applications [1]. 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 as common to 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 [2] and H.265 [3] allow for a dynamic rescaling in time (adaptive framerate) and quality (adaptive quantization). The scalable video coding extensions [4] enable additional spatial scaling and arrange layers in packet streams. All codecs need a proper parametrization for establishing a robust and reliable video conversation.

Appropriate video scaling is derived from estimating the 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.

In this paper, 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) A new 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 adaptation of the inter-arrival jitter for estimating network conditions at the receiver with an extra feedback channel to rescale the sender.
3) Experimental analysis of video adaption strategies based on the estimates of the sender, the receiver, and a combination of both.

We perform extensive experimental evaluations that include real-world 3G networks to analyse and justify our approaches. Our findings include an overall successful adaptation for each algorithm, with a faster reactivity of the sender-sided scheme, but a higher accuracy in bandwidth prediction at the receiver.

The remainder of this paper is organized as follows. Section II discusses the problems and requirements for an adaptive video conferencing software, followed by references to related work in Section III. Our core algorithms for detecting congestion and estimating appropriate scaling parameters is presented in Section IV. These approaches are applied and extensively evaluated with the help of our full-fledged video conferencing software system as documented in Section V. We conclude with an outlook in Section VI.

II. Problem Statement

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.

A. Requirements

In most video scenarios it is more important to provide a continuous video stream rather than a high resolution with interruptions. The time to identify congestion and estimate the bandwidth is crucial. As soon as a congestion is detected, the video stream needs downscaling to the available bandwidth.

Any solution that gives input to parametrizing the video codec should not introduce additional complexity to the network. Dedicated bandwidth probing techniques should be avoided as they increase congestion in large-scale deployments. A change of common network protocols should not be part of the solution space as this hinders deployment.

B. Discussion of the Solution Space

A common approach to determine a congested path based on video stream mechanisms is to observe the filling level of the receive buffer [5]. Unfortunately, this is not applicable in 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.

Who should control?—Observation 1: In the spirit of end-to-end connectivity, solutions should not be implemented at middleboxes, but at the sender or the receiver. The receiver does not control the encoding or the transmission of the video flow. Unlike the sender, it has no information about the transmission time, but can only measure time differences *and* the actually arriving bit rate. For a fast video adaptation, the sender should estimate and control the video quality directly. This is an advantage over a receiver-sided adaptation as it does not need an additional response channel to the sender and is thus faster and introduces less overhead.

Which measure indicates congestion?—Observation 2: Network overload fills router queues that cause delay. Its detection should avoid additional signaling, but be built on data already available from transport. Typically, transport protocols that maintain transmission state like TCP, Reliable UDP, or RTP/RTCP provide inherent information about the transmission time, usually in terms of the round-trip time (RTT).

However, a high RTT does not necessarily indicate a congestion. The RTT sums the one-way delays 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 [6]. For video stream transmission, only the one-way delay from the source to the destination is relevant, but difficult to measure. It requires synchronized clocks [7]—a complex task. It also produces overhead, especially in large-scale multimedia applications with multiple participants.

In summary, both sender and receiver can detect congestion and derive countermeasures. Most notably, the receiver can measure sustained bandwidths, frame rates and inter-arrival jitter, while the sender can measure RTTs and react immediately to prevent network overuse. Depending on the video scenario a sender-based, a receiver-based, or combining both may be appropriate. We summarize our findings in Table I.

TABLE I. OVERVIEW OF APPLICATION REQUIREMENTS AND INVOLVED PARTIES

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

III. 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.

1) Scaling a video stream in heterogeneous environments: Schierl *et al.* [8] present an overview of basic approaches to deploying scalable video in mobile realms, regarding various set-ups for IP and non-IP worlds. Particular focus is given 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. Insights are given on how the SVC interacts with mobile networks, different QoS metrics, and content delivery protocols. A real-world implementation of scalable mobile video conferencing is presented and analysed in [9], including first steps towards a dynamic network adaptation.

2) Bandwidth measurement: Common approaches to measure the available bandwidth such as the Probe Gap Model (PGM) or the Probe Rate Model (PRM) operate in parallel to the application by separate probing packets [10]. The accuracy of the bandwidth estimation depends on the probing frequency, the nature of side traffic, and the duration of the measurement. For a good comparative overview on these approaches including tools, we refer to [11], [12]. These techniques are intrusive, rather slow, and not compliant to the requirements for a real-time video conference.

3) Receiver-sided approaches: Barzuza *et al.* presented TREND [13], a receiver-sided approach to scale a video stream to network conditions. TREND was 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 receiver detects a delay by real-time monitoring of subsequent video frames. A frame is considered delayed, if the inter-arrival gap between two frames differs from the gap of the RTP timestamps. Basically, the algorithm raises the quality of the video stream until a delayed frame is detected at the receiver. Thereafter, the video stream is set to the last stable

bit rate prior to the congestion and starts increasing it slowly after some time. A similar approach is taken at the receiver side of the Google Congestion Control algorithm (GCC) [14] that is discussed in the RTCWeb context of the IETF.

Nguyen and Ostermann [15] examine video streams in an alternative scheme directly following a PGM approach. Their work is part of a streaming system, which uses the scalability extension of H.264/AVC and provides congestion control. Their approach is similar to the bandwidth estimation algorithm PTR [16], which is also a Probe Gap Model (PGM) approach. Instead of extra probing packets, the sending time of the RTP [17] video packets is manipulated for re-use as probing packets. Therefore, the RTP packets are sent with a predefined gap. Like in PGM, the competing traffic will influence the gap between the RTP packets.

Our approaches refrain from changing the semantic or use of transport protocols like RTP, but proceeds in a non-intrusive way by solely evaluating the arrival of regular video packets.

4) Sender-sided approaches: Precise one-way bandwidth estimations at the sender-side are complex, and common approaches follow a TCP-style approach by examining the loss rate, which is also influenced by the return path [18]. The measurement results are less accurate than receiver-sided measurements. On the other hand, loss can be used as congestion indicators like in the TCP protocol, introducing well-known problems in the wireless world. Such a sender-sided video adaptation approach is presented by Jang *et al.* [19], as well as the sender side of the Google Congestion Control algorithm (GCC) [14].

The IETF RTCWeb working group[1] is standarizing a set of protocols such as congestion control for real-time multimedia transport flows. One of the proposed approaches is the Google Congestion Control algorithm (GCC) [20]. The GCC is a TCP-friendly, real-time multimedia adaptation algorithm that uses UDP/RTP and is already implemented in some browsers. It consist of a receiver-sided and a sender-sided approach.

The receiver-sided approach uses the receive time of the frames to identify an over-used path. The difference in time of a frame and its predecessor is compared to the difference in their playout timestamps. A frame is delayed if the arrival time difference is larger than the timestamp difference. This occurs if the framesize varies or the sending rate exceeds the available bandwidth and queuing delays occur. The mean delay is observed and if it exceeds a certain threshold, the receiver indicates the path as congested and a downscaling of the multimedia stream is considered. If the detected congestion holds on for certain amount of time and frames, an 'over-use' of the path will be signaled to the sender. An 'under-use' of the link is signaled if the mean delay is below a certain threshold. If no 'under-use' nor an 'over-use' is detected, a 'normal' is signaled to the sender. The bit rate of the multimedia stream is increased by factor which is a function of the global system response time and the estimated measurement noise until a 'over-use' is signaled.

The sender-sided approach is based on packet loss, which is detected with frequently sent response messages from the receiver via RTCP. The RTCP reports include the fraction of lost packets and based on the amount, the sender-sided congestion control scales the multimedia stream. If 0-2% of the packets are lost, the bit rate is increased (a). If 2-10% of the packets are lost, the bit rate stays unchanged (b) and if more than 10% of the packets are lost, the bit rate gets downscaled (c). The sender-sided congestion control cannot scale the bitrate arbitrarily, but is limited by two rates. The bit rate cannot exceed the bandwidth estimation from the receiver-sided congestion control and it cannot be lower than the estimation of the TCP Friendly Rate Control formula [21].

[1] http://tools.ietf.org/wg/rtcweb/

TABLE II. NOTATION

Variable	Description
$\mu(t)$	Effective bandwidth on a path
$\beta(t)$	Bitrate of the video stream
$\omega(t)$	Frame rate of the video
$k(t)$	video scaling factor
d	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

A well known effect of a bandwidth adaptation based on the loss rate, is that they tend to fill the bottleneck queue. The filled queues delay the video flow, which is unwanted in real-time multimedia realms. However, the receiver-sided congestion is only reliable when the queue sizes along the paths are large and can hold enough packets to delay the video stream without losing packets. If the queues are short, packet loss occurs before the delays are visible in the receiver-sided congestion control. Thus, the packet loss based congestion control at the sender-side is important to make the approach reliable on paths with short queues.

5) Consideration of side-traffic: Streaming servers in the Internet are another application class with highly adaptive bandwidth demands and often use HTTP progressive download over TCP [22]. Huang *et al.* examined video bit rate adaptation strategies of three popular video streaming services (Hulu, Netflix, and Vudu) [23]. Even though these services operate over HTTP/TCP and adaptations compete with TCP flow control, their work yields fundamental insights into the problems which common video applications face when they have to compete with side-traffic.

6) Open issues: In previous work, we presented a preliminary implementation of a sender-sided congestion detection [24], which is elaborated and enhanced in the present paper. To the best of our knowledge, there is no generic sender-sided traffic adaptation scheme nor a detailed analysis comparing sender-sided, receiver-sided, and hybrid approaches.

IV. VIDEO CODEC ADAPTATION

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 bit rate $\beta(t)$, 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 bit rate matches the effective bandwidth, i.e., $\beta(t) * k(t) = \mu(t)$.

For scaling, it is not necessary to measure the available bandwidth μ directly, nor to know the current video bitrate, but it suffices to estimate the bandwidth ratio $k(t)$. 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. Table II summarizes the notation that we use for deriving the following algorithms.

A. 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. We assume a near packet-wise feedback from the receiver that can be harvested from stateful transport (see Section II). It is worth noting that RTCP feedback according to RFC 3550 is too slow, but rapid feedback mechanisms have been standardised in [25].

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

$$RTT(t) = d + q(t) \tag{1}$$

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

$$RTT'(t) = q'(t) \approx 0 \tag{2}$$

Less simplistic, 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

$$RTT''(t) = q''(t) \gg 0 \tag{3}$$

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.

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 detected congestion and the transmission of the next frame. The queuing occurs at routers that have an egress to a congested link and the queue process follows the rate equation

$$L(t_f) - L(t_c) = \int_{t_c}^{t_f} \beta(\tau) - \mu(\tau)\ d\tau, \tag{4}$$

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

$$\Delta q(t_c, t_f) \approx \frac{1}{\mu(t_i)} \int_{t_c}^{t_f} (\beta(\tau) - \mu(\tau))\ d\tau \tag{5}$$

As shown in (2), the queuing delay variations are also visible in the RTT jitter

$$\Delta RTT(t_c, t_f) \equiv RTT(t_f) - RTT(t_c) \approx \Delta q(t_c, t_f) \tag{6}$$

For a small interval, we assume the bit rate and the available bandwidth constant $\mu(t) = \mu,\ \beta(t) = \beta$, which resolves (5) combined with (6) to

$$\Delta RTT(t_c, t_f) \approx \frac{1}{\mu}(\beta - \mu) * (t_f - t_c) \tag{7}$$

This expression can be solved for k

$$k = \frac{\mu}{\beta} = \frac{(t_f - t_c)}{\Delta RTT(t_c, t_f) + (t_f - t_c)} \tag{8}$$

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.[2]

This approach predicts impending congestions in a very fast and light-weight way. 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.

B. A Receiver-sided Algorithm

The sender-sided approach operates fast and provides a scaled video stream that never exceeds the available bandwidth (see Section V). 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.

At the receiver side, the general idea is to compare the incoming bit rate with the ideally required bit rate of the video stream. If the incoming data rate at the receiver stays below the original bit rate of the video stream, a link is considered to be congested. Additional bandwidth is available and the video transmission can be upscaled, otherwise. As for the sender-side, 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 for measuring 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 for calculating the current bit rate for a frame received with length l

$$\beta(t) = l * \omega(t). \tag{9}$$

For an approximation of the available bandwidth, the receiver continuously measures the incoming bit rate of the video

[2]We will see in the next section that receiver-sided measures can correct erroneous early predictions by the sender.

stream, including the reception-time of the first and the last bit of a frame with n bits

$$\mu = \frac{l}{r_n(x) - r_1(x)} \tag{10}$$

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

Combining equations (9) and (10) yields the scaling factor k as predicted from the receiver side

$$k = \frac{\mu}{\beta} = \frac{\omega}{r_n(x) - r_1(x)} \tag{11}$$

This receiver-sided approach differs from the sender-sided approach by making use of complete frames for the measurement and is thus slower. Instead, it can accurately suggest upscaling and is capable of recognizing an underused link.

V. Performance Evaluation

A. Measurement Setup

1) Basic methodology: The Internet consists of many highly heterogeneous links carrying multiple competing traffic streams. The available bandwidth depends on both, the amount of UDP and TCP connections and traffic characteristics of the applications. For example, an 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 be sensitive to congestion and adapt the traffic depending on the available bandwidth. Emulating this variety on a very fine-grained level is difficult and usually does not help to highlight specific protocol effects. 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, as well as on a deployment in a real 3G network.

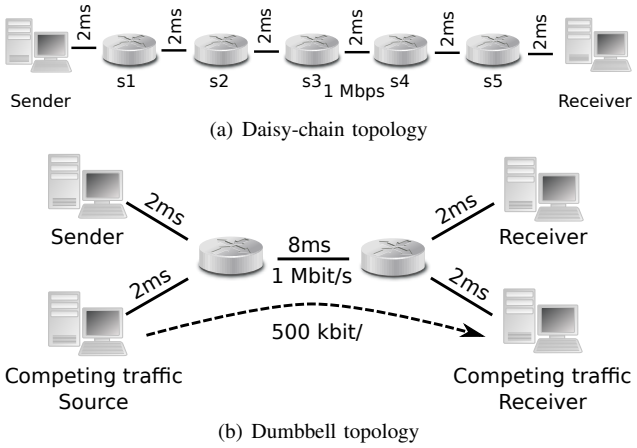
2) Environment: For the emulation measurements, we used Mininet[3]. Mininet is an open source network emulator, which can be used to create virtual networks with controlled network conditions. To gain ground truth in our results, we also verified the measurements by running selected experiments in a real network. Deployment in real-world mobile networks was undertaken by using an UMTS uplink from a mobile sender.

3) Topologies: We used two basic test topologies for the networks in our performance evaluation (cf., Figure 1), i.e., a daisy-chain topology and a dumbbell topology, before probing in real-world mobile networks. 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 500 kbit/s competing traffic stream. The competing traffic is emulated by iperf[4].

In both topologies, the emulated bandwidth is limited to 1 Mbps, and the RTT between video sender and receiver is set to 24 ms. This is considered to be a short RTT network [26]. The one-way delay for all (full-duplex) links has been adjusted accordingly, i.e., either 2 ms or 8 ms (cf., Figure 1). 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. This means that the signalling between sender and receiver can be completed before the next frame is encoded in the receiver-sided adaptation approach. In the 3G mobile production networks, the available bandwidth varied between 1 Mbit/s and 1.7 Mbit/s. More significantly, RTT values from 60 ms to above 1 s were observed on a free link with an average variation of the RTT around 60 ms.

[3]http://mininet.org

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



(a) Daisy-chain topology

(b) Dumbbell topology

Fig. 1. Measurement setup

4) Video sequences and video codec: To evaluate our approaches 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 we present only the results of the TW test sequence in the following subsections. In Section V-G we show a comparison of all test sequences.

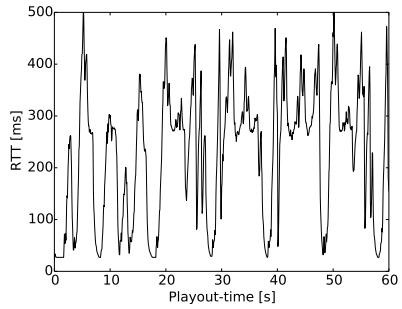
All measurements are conducted using the DSVC codec [9], which supports up to three temporal layers. This is an extension of the very efficient DAVC H.264/AVC implementation. The codec is used in commercial products and thus complies with real-world requirements.

5) Metrics: We evaluate the performance of the adaptation approaches based on the metrics RTT, jitter variation, bit rate and video quality, and the distribution of the inter-arrival jitter. For easy comparison between the different experiments, we always sample inter-arrival jitter in bins of 2 ms.

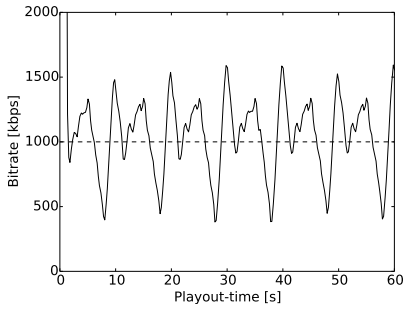
In the remainder of this section, we first present the performance of an unscaled video, which clearly motivates the need for scalable adaptive video coding. We compare sender-sided, receiver-sided, and hybrid adaptation in the second, followed by the mobile deployment in a 3G network.

B. Unscaled Video Stream

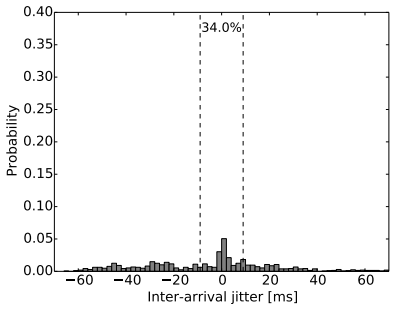
The video source sends an *unscaled* version of the TW test sequence with the highest quality, i.e., a maximal quantization factor is configured and all three temporal layers are enabled.



(a) RTT measured at the sender

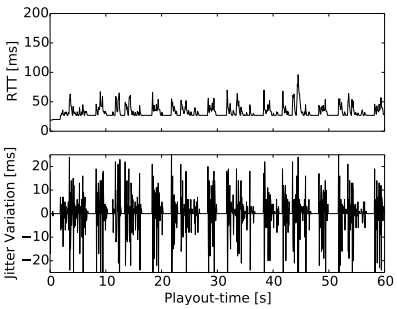


(b) Bitrate of the video stream

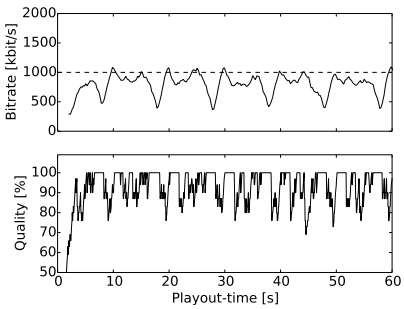


(c) Inter-arrival jitter measured at the receiver

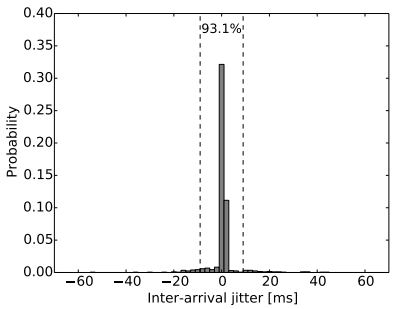
Fig. 2. Unscaled video stream with a maximal bandwidth of 1000 kbps



(a) RTT and jitter measured at the sender



(b) Bitrate and codec adaptation of the video



(c) Inter-arrival jitter measured at the receiver

Fig. 3. Sender-sided video adaptation

The bitrate of the unscaled video stream varies between 500 kbit/s and 1.5 Mbit/s (cf., Figure 2(b)). The available bandwidth of 1 Mbit/s is clearly exceeded. The resulting congestion influences the RTT significantly (cf., Figure 2(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 video conferencing applications the maximal one-way delay should be around 100 ms to not distract end users [27].

As a consequence of the congestion frames do not arrive in time. Figure 2(c) shows the distribution of the inter-arrival jitter per frame at the receiver. The huge variations in the inter-arrival jitter are perceived by the user as a stuttering video stream. The Media Delivery Index (MDI) [28] 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 34 % of the frames fulfill these requirements; appropriate video conferencing is not possible.

In the next sections, we show how an adaptive video codec can cope with congestions. Our objective is to provide a fluent video stream. We use the 9 ms inter-arrival jitter from the MDI as reference to judge about the applicability in real-world deployment.

C. 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 was detected. Until the sender explores congestion, we use a very aggressive upscaling strategy and continously increase the quality after a frame was sent.

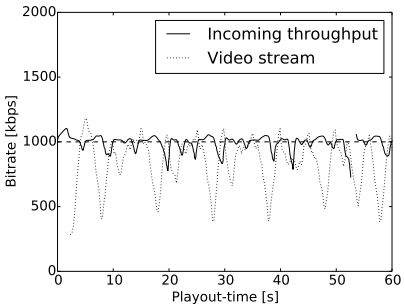
Figure 3(a) shows the RTT and the jitter variation of the RTT, which is used to indicate a congested link. Every increasing of the RTT goes along with a significant jump in the jitter variation, which triggers a downscaling. Figure 3(c) shows the inter-arrival jitter distribution of the frame reception on the receiver-side. Compared to the unscaled video stream, 93.1% of the frames arrive in time with at most 9 ms jitter.

Figure 3(b) shows the resulting bitrate of the video stream and the codec quality adaptation. At the beginning, the video stream starts with the worst quality settings, but increases rapidly. Every detected congestion that increases the RTT follows an adaptation to the estimated available bandwidth. The encoding quality varies between 75% and 100%, which corresponds to quantization factors below 25. The video thus remains at high quality [29]. As result, the bitrate of the video stream stays around or below the available bandwidth and does not congest the link.

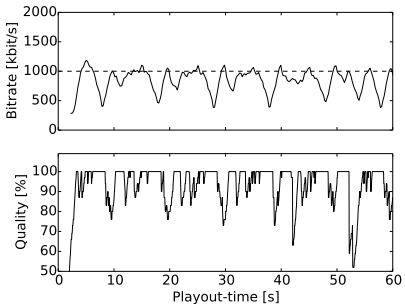
D. Receiver-sided Video Adaptation

In this scenario, only the receiver scales the video stream. The receiver-sided congestion control compares the incoming rate with the current bitrate of the video stream (cf., Figure 4(a)). The incoming throughput varies around 1 Mbit/s. As soon as the bitrate of the video stream is larger (or smaller), the receiver signals a downscaling (or upscaling) to the sender.

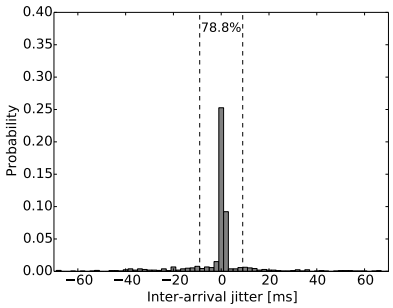
Figure 4(b) shows the bitrate and the codec adaptation of the rescaled video stream. Both measurements exhibit similar



(a) Incoming throughput and bitrate of the video stream

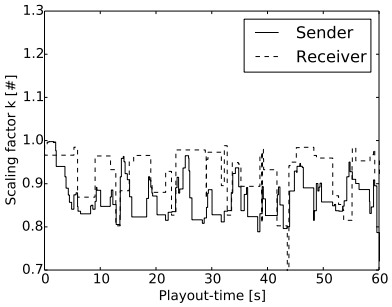


(b) Bitrate and codec adaptation of the video

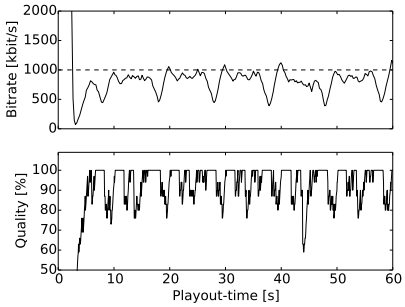


(c) Inter-arrival jitter measured at the receiver

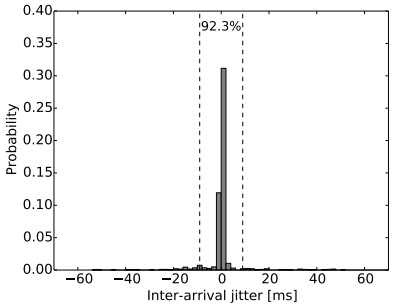
Fig. 4. Receiver-sided video adaptation



(a) Quality change suggestions for the codec adaptation



(b) Bitrate and codec adaptation of the video



(c) Inter-arrival jitter measured at the receiver

Fig. 5. Sender-sided and receiver-sided video adaptation

behaviour compared to the sender-sided codec adaptation approach. The receiver-sided adaptation is less sensitive to RTT fluctuations, i.e., over a longer time frame the change of the video quality is less frequent. However, amplitudes are larger.

The distribution of the inter-arrival jitter at the receiver is shown in Figure 4(c). A fairly stable video stream transmission is clearly visible. 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 78.8 % of the frames arrive in due time.

E. Sender-sided and Receiver-sided Video Adaptation

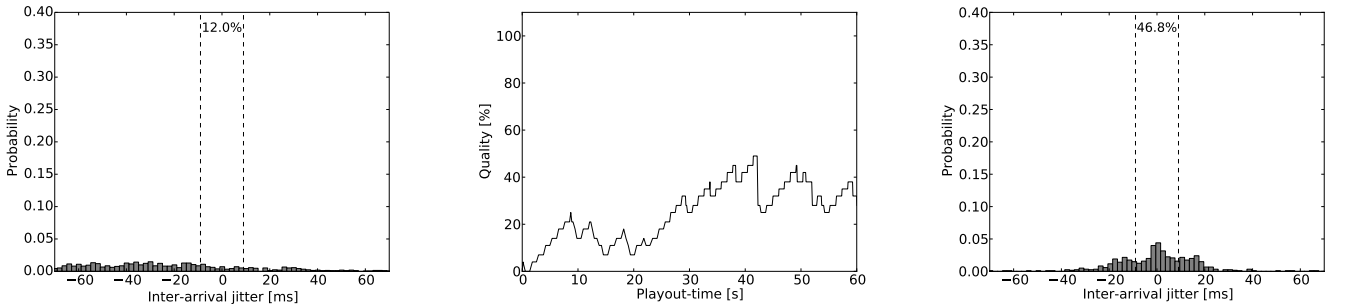
In this scenario, the receiver *and* the sender influence the scaling of the video stream. Both can initiate a downscaling of the video quality (i.e., reduce the bit rate), while only the receiver is allowed to increase the video quality.

The sender and the receiver calculate a scaling factor k. Figure 5(a) visualizes the evolution of this value for both parties over time. The sender makes more downscaling requests which are less persistent than the receiver-sided requests. This complies with our previous observations, which showed that the sender reacts more sensitive to network changes (cf., Section V-C and Section V-D). Most of the time, the sender reacts fast enough to avoid any congestion that could also be detected by the receiver. There are two (rare) cases where the receiver-sided adaptation compensates the sender-sided approach. (a) The measurement period of the sender is too short and thus lacks 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 downscaling is requested by the sender, the codec adaptation looks very similar to the pure sender-based adaptation (cf., Figure 5(b)). Compared to the unscaled stream, the scaled video exhibits still a high quality and varies between 70% and 100%, which ensures a good viewer experience. The bit rate stays around or below the available bandwidth and the video stream is adapted reliably. In general, the results show a fairly stable video transmission with just a few disturbances. Strikingly, 92.3 % of the incoming frames exhibit a jitter below 9 ms (cf., Figure 5(c)).

F. Measurements in Real-World 3G Networks

3G wireless networks introduce significant delays and alienating delay variations in the absence of congestions as discussed above. In an unscaled video, this leads to a disgraceful performance of only 12 % of the frames remaining within the 9 ms jitter bound (cf., Figure 6(a)), as well as unpredictable algorithmic fluctuations. The video quality flaps around appropriate values and the application is never able to reach a high quality stream. This can be improved by a temporal blocking of upscaling after each re-scaling. In detail, we set a constant timer of 1 s in our scenario for this delay. The temporal blocking slows down the upscaling process, but sustain a robust and stable video stream.



(a) Inter-arrival jitter distribution of the unscaled video stream

(b) Codec adaptation of the video

(c) Inter-arrival jitter distribution of the scaled video stream

Fig. 6. Scaling analysis for video streams in a real-world UMTS network

The distribution of the inter-arrival jitter of a scaled video stream in Fig. 6(c) depicts an improvement up to about 46.8 %. In contrast to the previous measurements, increasing the quality of the video is more cautious and shows a stable and converging course (cf., Fig. 6(b)).

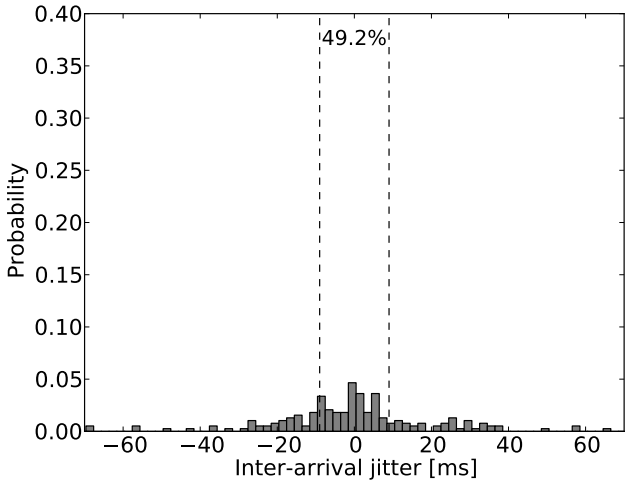


Fig. 7. Inter-arrival jitter distribution for a video stream with an inter-arrival based video scaling at the receiver-side

The same test configuration is also tested with an inter-arrival based scaling approach at the receiver-side, which is a widely used technique to scale video streams [14]. The measurement results are slightly better with 49.2% in due time arriving frames (cf., Figure V-F), but it is only a minor improvement. In both scenarios, the sender-sided approach reacts faster and both receiver-sided algorithms sparely detect a congestion. Therefore, the measurements results are only slightly influenced by the receiver-sided algorithm.

In summary, our experiments validated that the proposed approach to sender-sided video scaling could approximate the difficult transmission regime of UMTS packet transmissions in real-world networks.

TABLE III. INTER-ARRIVAL JITTER BELOW 9 MS

Video	Unscaled	Scaled
TW	34.04%	92.31%
G4	66.38%	97.33%
KO	20.32%	81.61%
SM	75.90%	96.39%
TC	22.87%	88.56%
UH	0.55%	92.14%
Elephants Dream	0.00%	92.00%

G. Comparison of the Test Sequences

Finally, we compare the quality of the sender-sided and receiver-sided approach by deploying all six test sequences in the daisy chain topology. Table III shows the percentage of frames that complies with the Media Delivery Index, i.e., a high quality video streaming requires an inter-arrival jitter below 9 ms. Overall, scalable video coding results in a huge improvement for all test sequences. Varying values show that characteristics of the video significantly influence the effects of scaling: By comparing the test sequence ”UH” with ”TC”, it becomes evident that the characteristic of the video largely influences scalability. The testsequence ”UH” constantly attains high bit rates, while the testsequence ”TC” has a low and fluctuating bit rate.

The previous test scenarios uses short video sequences and thus, the measurement time was very short and allows a 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 tested with longer video sequences. The movie Elephants Dream[5] is used in this measurement. Without scaling, the path congests within the first minute and never recovers. After a few minutes the connection disconnects. The scaled video stream reaches the available bandwidth and rarely exceeds it. Overall, the scaling is reliable and the provided video stream has a high quality considering the network conditions. 92% of the frames arrive in time and the user perceive only minor impairments.

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

VI. Conclusion

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, we presented a combination of a receiver-sided and a sender-sided congestion detection approach combined with scalable video adaptation 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 on 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 for 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 adds reliability.

In future work, we will transfer this adaptation scheme to PlaceCam [6], our professional video conferencing system, and evaluate performance statistics from real-world deployment on a large scale. Analyzing various video samples and different network conditions will help to ensure reliable video adaptation even in challenging environments.

References

[1] H. L. Cycon, T. C. Schmidt, G. Hege, M. Wählisch, D. Marpe, and M. Palkow, "Peer-to-Peer Videoconferencing with H.264 Software Codec for Mobiles," in *WoWMoM08 – 9th IEEE Intern. Symposium on a World of Wireless, Mobile and Multimedia Networks – WS Mobile Video Delivery (MoViD)*, R. Jain and M. Kumar, Eds., IEEE. Piscataway, NJ, USA: IEEE Press, June 2008, pp. 1–6.

[2] "Advanced Video Coding for Generic Audiovisual Services," ITU-T, Tech. Rep. Recommendation H.264 & ISO/IEC 14496-10 AVC, v3, 2005.

[3] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (hevc) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649 –1668, December 2012.

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

[5] F. Wamser, D. Hock, M. Seufert, B. Staehle, R. Pries, and P. Tran-Gia, "Using buffered playtime for QoE-oriented resource management of YouTube video streaming," *Trans. Emerging Tel. Tech.*, vol. 24, no. 3, pp. 288–302, April 2013.

[6] S. Kaune, M. Wählisch, and K. Pussep, "Modeling the Internet Delay Space and its Application in Large Scale P2P Simulation," in *Modeling and Tools for Network Simulation*, K. Wehrle, M. Günes, and J. Gross, Eds. Heidelberg: Springer, 2010, pp. 427–446.

[7] L. De Vito, S. Rapuano, and L. Tomaciello, "One-way delay measurement: State of the art," *Instrumentation and Measurement, IEEE Transactions on*, vol. 57, no. 12, pp. 2742–2750, December 2008.

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

[9] H. L. Cycon, T. C. Schmidt, M. Wählisch, D. Marpe, and M. Winken, "A Temporally Scalable Video Codec and its Applications to a Video Conferencing System with Dynamic Network Adaption for Mobiles," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 3, pp. 1408–1415, August 2011.

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

[11] E. Goldoni and M. Schivi, "End-to-end available bandwidth estimation tools, an experimental comparison," in *Proceedings of the Second international conference on Traffic Monitoring and Analysis*, ser. TMA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 171–182.

[12] C. D. Guerrero and M. A. Labrador, "On the applicability of available bandwidth estimation techniques and tools," *Comput. Commun.*, vol. 33, no. 1, pp. 11–22, 2010.

[13] 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*, 2010, pp. 126–133.

[14] S. Holmer, L. D. Cicco, S. Mascolo, and H. Alvestrand, "A Google Congestion Control Algorithm for Real-Time Communication," IETF, Internet-Draft – work in progress 02, February 2014.

[15] D. T. 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*, vol. 1, no. 2, pp. 246 –253, August 2007.

[16] N. Hu and P. Steenkiste, "Evaluation and Characterization of Available Bandwidth Probing Techniques," *IEEE Journal on Selected Areas in Communications*, vol. 21, pp. 879–894, 2003.

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

[18] 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 Intern. Conference*, 2013, pp. 529–534.

[19] E.-D. Jang, J.-G. Kim, T. C. Thang, and J.-W. Kang, "Adaptation of Scalable Video Coding to packet loss and its performance analysis," in *Advanced Communication Technology (ICACT), 2010 The 12th International Conference on*, vol. 1, 2010, pp. 696–700.

[20] S. Holmer, L. D. Cicco, S. Mascolo, and H. Alvestrand, "A Google Congestion Control Algorithm for Real-Time Communication," IETF, Internet-Draft – work in progress 02, February 2014.

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

[22] R. Pantos, "HTTP Live Streaming," IETF, Internet-Draft – work in progress 13, April 2014.

[23] T.-Y. Huang, N. Handigol, B. Heller, N. McKeown, and R. Johari, "Confused, Timid, and Unstable: Picking a Video Streaming Rate is Hard," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 225–238.

[24] F. Jäger, T. C. Schmidt, and M. 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: IEEE Press, Sep. 2012.

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

[26] S. Ha, I. Rhee, and L. Xu, "CUBIC: a new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, July 2008.

[27] ITU, "G.114 - One-way transmission time," ITU, Recommendation - Telecommunication Union Standardization Sector, 05 2003.

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

[29] Y.-F. Ou, Z. Ma, T. Liu, and Y. Wang, "Perceptual quality assessment of video considering both frame rate and quantization artifacts," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 21, no. 3, pp. 286–298, March 2011.

[6] http://www.daviko.com/videokonferenz/22-1-PlaceCam-3.html