Evalvid-RA: Trace Driven Simulation of Rate Adaptive MPEG-4 VBR Video

Arne Lie†and Jirka Klaue‡
†SINTEF ICT, Dept. of Communication Systems, Trondheim, NORWAY,
arne.lie@sintef.no.
‡Technical University of Berlin, Telecommunication Networks Group, Berlin, GERMANY,
jklaue@tkn.tu-berlin.de

The original publication is available at www.springerlink.com.
Direct link: http://dx.doi.org/10.1007/s00530-007-0110-0

***Abstract*— Due to the increasing deployment of conversational real-time applications like VoIP and videoconferencing, the Internet is today facing new challenges. Low end-to-end delay is a vital QoS requirement for these applications, and the best effort Internet architecture does not support this natively. The delay and packet loss statistics are directly coupled to the aggregated traffic characteristics when link utilization is close to saturation. In order to investigate the behavior and quality of such applications under heavy network load, it is therefore necessary to create genuine traffic patterns. *Trace files* of real compressed video and audio are text files containing the number of bytes per video and audio frame. These can serve as material to construct mathematical traffic models. They can also serve as traffic generators in network simulators since they determine the packet sizes and their time schedule. However, to inspect perceived quality, the compressed binary content is needed to ensure decoding of received media. The EvalVid streaming video tool-set enables this using a sophisticated reassembly engine.**

Nevertheless, there has been a lack of research solutions for *rate adaptive* media content. The Internet community fears a congestion collapse if the usage of non-adaptive media content continues to grow. This paper presents a solution named Evalvid-RA for the simulation of true rate adaptive video. The solution generates real rate adaptive MPEG-4 streaming traffic, using the quantizer scale for adjusting the sending rate. A feedback based VBR rate controller is used at simulation time, supporting TFRC and a proprietary congestion control system named P-AQM. Example *ns*-2 simulations of TFRC and P-AQM demonstrate Evalvid-RA's capabilities in performing close-to-true rate adaptive codec operation with low complexity to enable the simulation of large networks with many adaptive media sources on a single computer.

***Index Terms*— Congestion control, rate control, streaming media, VBR video, network simulation.**

I. Introduction

THE Internet is today facing a change of the traffic type dominating the aggregates at network core and edges. Interactive VoIP and videoconferencing are currently having an exponential growth of usage, but also one-way streaming media (e.g. VoD and WebTV) is experiencing large growth rates [1], [2]. Since the majority of this media content is controlled by technology that does not monitor traffic load nor scale the bit rate during the ongoing sessions, serious quality degradation due to traffic overload (i.e. packet drops and excessive delays) and throughput unfairness might result. Typically, such services probe the network throughput only during session startup, if at all, and initiates one of a few possible quality versions based on current network state and end user terminal characteristics. The MPEG and commercial video communities have developed several advanced solutions to answer the media scalability challenge: (i) scalable video with base layer and enhancement layers [3], (ii) FGS (Fine Granular Scalability) [4], and (iii) several multi-rate coding schemes (e.g. Envivio, Microsoft Intelligent Streaming, Real SureStream). (i) has the benefit of efficient file storage, but the total flow sent has lower compression efficiency than flows from codecs with only a single layer. FGS can be adjusted to finer bandwidth granularity than ordinary scalable coding, at the cost of higher complexity and still lower coding efficiency [5]. While MPEG-4 FGS has failed in the market, the new H.264 SVC might give FGS related technologies a new chance [6].[1] Multi-rate coding stores typically three tracks with different optimized bit rates in a single file, and the selected track can be switched on-the-fly during streaming time. While suffering from the highest storage capacity needs, this solution is still receiving most commercial interest, due to its simplicity and good transmission bandwidth utilization.

In contrast to offline coding approaches, which actually build their scalability capabilities on coarse network state assumptions, *online real-time codecs* for e.g. videoconferencing can adjust codec parameters on the fly to adapt to the current network state on much finer time granularity. This paper presents analysis and tools supporting research within real-time encoding, but as our conclusions will argue, its architecture is applicable also within offline encoding.

If popular media continues to be non-adaptive, video services may consume much more than their fair share of capacity such as when competing with TCP flows at network bottlenecks, and this breaks the best effort Internet[2] principles [7]. This unfairness adds to the already mentioned problems with queuing delay and packet loss. As an answer to these network challenges, the IETF has during the last years worked on a new real-time media transport protocol named Datagram Congestion Control Protocol (DCCP) [8], to support the deployment of rate adaptive codecs. UDP has

[1] Still, FGS is very complex and cannot be part of real-time encoding.

[2] Even this paper focus on best effort class of traffic, rate adaptation can also be used within DiffServ classes. Note that also expedited forwarding DiffServ QoS breaks if too many non-adaptive applications are requesting it.

no congestion control mechanism like TCP. The main idea of DCCP is to continue to use UDPs non-reliable packet flow (no retransmissions in case of packet drops), but make it connection-oriented like TCP. The latter will enable better firewall penetration capabilities and the possibility to exchange different parameter values at session initiation, such as the choice of congestion control algorithm. TFRC (TCP Friendly Rate Control) is the most fitted DCCP congestion control profile for video traffic [9], using equation based control in order to obtain smooth rate at an average similar to that of TCP.

Many other solutions have also been proposed over the last decade to solve these problems, among them VBR over ATM ABR services [10], RAP [11], MPEG-TFRCP [12], LDA+ [13], and P-AQM+ECF [14]. All of these have slightly different objectives, but agree on the target goal of assisting the network to provide fair and stable services. The proposals can be divided into two main groups: (i) those who are pure end-to-end oriented and only monitor the network state by packet loss statistics feedback, and (ii) those who in addition also take advantage of more advanced network state information, such as the binary ECN marks [15], or explicit information on traffic load from each node on the path from sender to receiver. Since the Internet community puts a strong focus on scalability, pure end-to-end oriented systems are preferred. However, there are concerns whether this is sufficient to ensure low delay and packet loss in traffic overload situations. Vital parameters are rate adjustment speed and accuracy. The interplay with the other media delivery chain functionalities are also of major importance, such as traffic shaping, jitter buffer dimensioning and control, and decoder robustness to packet losses. The proposed solutions should therefore be compared taking all these parameters into account.

In order to perform research on vital streaming media parameters, both at network/transport layer *and* application layer, the setup of true multimedia test networks might seem necessary. This can however be very expensive and of little flexibility. Thus, network simulations, using tools like the *ns*-2, might seem tempting. The problem with the latter is that one is stuck with either using synthetic video/audio models or static audiovisual trace files for source traffic generation. Since our goal is to implement media rate control based on traffic feedback, the source models need to be rate adaptive. Such modification of synthetic models is straight forward, but then the goal of investigating perceived quality is excluded. For this reason real audiovisual trace files must be used in order to inspect perceived quality. One possibility for support of the latter is to use the EvalVid tools-set [16] invented by J. Klaue. EvalVid is an open-source project, and supports trace file generation of MPEG-4 as well as H.263 and H.264 video. Using it together with the *ns*-2 interfacing code suggested by C.-H. Ke [17], perceived quality and objective measure like PSNR calculation can be obtained after network simulation. But still, this does not provide a solution for *rate adaptive* video investigation.

All this has motivated the design and implementation of Evalvid-RA, a tool-set for rate adaptive VBR video investigation in *ns*-2, based on modifications to the EvalVid version 1.2 tool-set and the *ns*-2 interfacing code. The solution framework is generic so that it can be implemented within any network simulator, and on any codec, provided that a set of guidelines is followed. The paper is organized as follows: Section III gives first an overview over the standardized methods for video evaluation. In Section IV the necessary framework building blocks are introduced and explained. The performance of this framework is investigated in this paper using a video rate controller presented in Section V. By running *ns*-2 simulation example scenarios presented in Section VI, the Evalvid-RA capabilities are demonstrated, focusing on traffic characteristics and rate controller performance in various protocol and network environments. The contributions of this paper compared to referenced work are

- The EvalVid v1.2 tool-set is enhanced to support *rate adaptive* video (Evalvid-RA).
- The SVBR [18] rate controller is modified to become an adaptive rate controller (RA-SVBR).
- The absence of Long Range Dependency (LRD) in aggregate VBR rate adaptive video traffic without the use of traffic shaping buffer is demonstrated using Evalvid-RA's realistic video traffic generators.
- The quality of rate adaptive MPEG-4 streaming with conversational delay constraints is calculated using the Evalvid-RA tool-set (e.g. PSNR). Different protocols (UDP and TFRC) and network types (FIFO, RED, P-AQM) are used and compared in mixed TCP traffic scenarios.

The goal of this paper is to present the Evalvid-RA architecture, to validate its performance, and lastly to exemplify utilization in adaptive streaming media research, showing how increased network intelligence can improve streaming performance.

II. Related Work

Evalvid-RA connects multiple independent research areas: (i) media rate control, (ii) media traffic characteristics, (iii) network congestion control, and (iv) efficient and error resilient coding. Rate control includes sender and receiver buffer dimensioning, to avoid both overflow and underflow, as thoroughly analyzed in [19]. VBR video traffic characteristics have been reported e.g. by [20], [21] with following Markov and ARIMA modeling by e.g. [22]–[24]. The latter modeling however does not take adaptive rate control into account. Network congestion control schemes for media content were listed in the Introduction. Since Evalvid-RA includes real media decoding, coding efficiency and packet loss resiliency will also be taken into account, as PSNR and possibly other QoS measures are calculated, or decoded video is actually consumed by human observers.

In recent years, papers have been published on the topic of the simulation of rate adaptive media, and also real experimental studies have been set-up to test e.g. early DCCP prototypes efficiency. In [25] MPEG-4 trace files are used to calibrate a TES (Transform Expand Sample) mathematical model, and rate adaptation is incorporated by adjusting the frame size output by a scalar (from rate-distortion curve).

The simulation model however has no on-line rate controller, and since the traffic is synthetic, perceived quality cannot be investigated. In [26] the authors set up a simulation scenario where both temporal and quantizer scale adaption is possible. But again the traffic is synthetic. H.263 video trace files are used in [27], and the sending rate is controlled by DCCP TCP-like. However, the video is not rate adaptive, so the video submission is controlled by overruling the real-time constraint. In [28] models are derived for pre-recorded media streaming over TFRC and compared to simulations. The models focus on the impact of the TFRC rate changes to the probability of rebuffering events, i.e. events where the receive buffer is emptied. Recently, more realistic simulation implementations have been published, such as [29] where rate adaptation using frame discard and FGS has been studied and implemented in ns-2 by also inserting the binary content directly into the simulator packets, thus supporting media decoding and PSNR calculation. The benefit of inserting the binary data into the ns-2 packets is that there is no need of keeping track of additional simulation time trace files. However, the penalty is higher computational load at simulation time, limiting the practical size of the network and number of simultaneous video sources. [30] is an example of a recent experimental study of real VoIP traffic using DCCP, using real applications and networks.

To the best of our knowledge, Evalvid-RA is the first tool to create realistic "online" rate adaptive streaming media traffic. It includes

- a simulation time rate controller to modulate the quantizer scale used by a real codec
- realistic frame packetizing
- the ability (through ns-2) to choose network complexity, protocol and queue management support
- a framework that is scalable to a large number of simultaneous video sources
- and finally at the receiver side being able to restore the media files supporting PSNR and other QoS metrics calculation.

Due to the trace file approach of Evalvid-RA, absolute delay and delay jitter impairments to the media decoding process can be investigated in a post-process, thus decoupling network and receiver media player constraints. Although we recognize the importance of mathematical models for traffic and queue statistics analysis, we believe that the complexity of the heterogeneous networks makes realistic simulation a better tool, especially when being able to compute end-user QoS metrics such as PSNR, or even perform human subjective tests.

III. Video Quality Evaluation

The quality of a video transmission depends on the impression a human observer receives of the delivered video. Though traditional network metrics such as bandwidth, packet loss, jitter and delay, certainly influence the video quality, the perceived *subjective* quality impression of a human observer is nevertheless the most important factor. The subjective video quality test results are expressed by means of e.g. the mean opinion score (MOS) as defined by the ITU. The MOS is a scale from 5 (excellent) to 1 (bad). In contrast, *objective* video quality metrics are calculated by computers. Basically, these can be divided into pixel-based metrics, like SNR or *PSNR*, and *psycho-visual* metrics. The latter approach, which is based on models of the human visual system (HVS), has been shown to outperform standard quality metrics like PSNR in most cases [31], [32][3]. However, sometimes the absolute value of the video quality and its correlation to subjective tests is not the most important factor but rather the relative quality regarding a certain optimum. An example would be the comparison of different transport protocols with an assumed error-free transmission. In these cases simple metrics like PSNR are still adequate. Another downside of psycho-visual metrics is their complexity and, thus, huge computational overhead compared with PSNR.

If the influence of network characteristics and parameter optimization is to be assessed in terms of real subjective video quality, a dedicated metric should at least be included in the target function of the optimization. We recommend the application of the video performance estimation method standardized by ANSI [33], since it outperforms PSNR and similar methods as shown in e.g. [31]. Though a lot of research about video quality assessment has been done – and is still in progress – the field is by no means finished. Nevertheless a variety of reasons has been identified why objective metrics like PSNR are not adequate for performance evaluation. In [34] the influence of the frequency and amplitude of quality fluctuations in layered video transmission has been investigated. It has been shown (amongst others) that it is better to minimize the frequency of fluctuations even if the average PSNR decreases.

Another problem which must be faced is the quality assessment of long video sequences. Usually one quality indicator per video sequence is calculated, which describes the impression of an average (non-expert) human observer. This is well fitted for the relatively short video sequences for which these metrics are verified. However, one quality indicator is not enough for longer video sequences since short but sharp disturbances could be masked by the averaging over longer time spans. Since periodically occurring disturbances could influence the overall impression of a video transmission, a quality assessment method should also reflect this. One possible solution is the calculation of the video quality – with any method – in a sliding window of, e.g., 10 seconds. The quality indicator of each window is compared to the quality indicator of the corresponding video part before transmission. The frequency of degradations could be used as overall quality measure for the transmission instead of the averaged quality indicator. Another possibility is the specification of a threshold for a tolerated number of quality indicator deviations. The Evalvid-RA tool-set provides a method which can calculate these figures for long videos. For this purpose the `miv.exe` tool from EvalVid v1.2 is used. This quality indicator is introduced and explained in detail in [35].

There is no generally accepted method to access the quality of a video transmission system. Though some aspects of the problem have been discussed in this section, an in-depth study

[3] I.e. their results come closer to subjective tests.

would be beyond the scope of this paper. The citations in this section provide a good start for further reading.

The Evalvid-RA framework supports the use of any metric since the calculation of actual quality values is separated from the simulation process. Only PSNR-calculation is included directly in the tool-set, but the use of subjective metrics has successfully been tested in [36], [37] and [38]. The included MOS calculation tool uses a simple mapping of PSNR values to MOS (defined in [39]) which nevertheless achieves quite good correlation with [33] in most cases.

IV. The Evalvid-RA architecture guidelines

An efficient tool-set for network simulation must be scalable so that even large networks with many sources and many network nodes can be simulated on a single computer. Two major challenges result from this ambition: (i) perceptual quality inspection at receiving nodes, and (ii) the implementation of an adaptive rate controller having access to both media content and network state feedback. The first challenge could easily be solved by using real binary packet data as packet payload in the network simulator. However, such an attempt will degrade the simulator performance significantly. A more efficient approach is to use unique packet identifiers to support video frame assembly as a post-process. The existing EvalVid tools [16], [17] uses this approach, by introducing a trace file generation process, a network simulator process, and a post-process. The second challenge is however in conflict with the division between pre-process and network simulation process, because it is only the pre-process that has access to the media and codec itself. Thus, one need to find a method supporting the exchange of necessary information between those two processes. Obviously, the solution is dependent of the kind of rate controller in use.

A. The selection of a rate controller

Traditionally, video rate controllers are divided into three categories: (i) constant bit rate (CBR), (ii) variable bit rate (VBR), and (iii) quality based (open loop VBR). In CBR, the rate controller constraint is to produce a constant number of bits per time unit such as the Group of Pictures (GOP) (if it has a constant number of frames per GOP). To achieve this goal for a hybrid codec using DCT transform (e.g. MPEG), the *quantizer scale* (which holds the quantization value matrix for the DCT transformed 8x8 pixel blocks) is considered changed for each macro-block (16x16 pixel block) [40]. [4] The bit rate budget is optimized looking at several sequential video frames, causing an algorithmic delay in the rate controller. Due to this delay, interactive applications are better off with a VBR rate controller, which trades lower delay for higher bandwidth variability. Other benefits with VBR are more constant quality and higher multiplexing gain potential. VBR typically considers quantizer scale changes at each new video frame, or even only at each GOP. The third option, open loop VBR, is actually coding without any rate controller, i.e. the quantizer scale is fixed during the whole sequence[5], thus producing the highest bit rate variability. The bit rate produced is highest in high motion scenes, and when there are many details and hard contrasts.

To limit the size of the trace files needed as input to the network simulation, they are captured at frame granularity, i.e. the size of each frame in bytes is stored in a log file. This rules out CBR, since in that case we would have needed access to sizes on macro block granularity[6]. The rate controller choice will therefore be based on VBR. Before deciding on the granularity of the rate controller, the interplay between the pre-process and network simulator must first be considered.

B. The pre-process

The goal is to have an online rate controller in the network simulator, but without having to do the media encoding itself, since that will demand too much CPU resources during simulation time. The media encoding must be performed in a pre-process. In MPEG-4 [4], the valid quantizer scale values are in the range 1 to 31, with 1 producing the highest quality and bit rate. The key idea is then *to encode the media with open loop VBR for all possible quantizer scales*[7]*, store the frame sizes per quantizer scale in separate files, so that the online rate controller in the network simulator can select a new quantizer scale value and get the correct frame sizes from the corresponding trace file.* The simplest and most correct option is to allow the rate controller to consider a new quantizer scale only at the start of a new GOP. By keeping the GOP size fixed, the rate controller will always find an I-frame as the first frame after trace file switching. The concept is depicted in Fig. 1 for GOP size of two frames, and only three different quantizer scale values 2–4. The synchronized GOP boundaries will ensures a refresh of the motion prediction algorithm, and all succeeding P- and B-frames in that GOP will be based on that I-frame.

Changing to another quantizer scale during a GOP is however also possible without causing too noticeable artifacts, but a real encoder with rate controller would then produce the next P- or B-frame based on a slightly different compressed I-frame (i.e. the same frame but not the same quantizer scale) than the one used in the simulation. To explain this with an example, let's consider a codec that produces 12 frames per GOP and only I- and P-frames (the I-frame is number 1, while frames 2–12 are P-frames). The rate controller has chosen quantizer scale 5 for an ongoing GOP. At frame number 7 in that GOP, the rate controller suggests changing to scale 10, since the bit rate budget is somewhat overrun. A real live encoder would then produce frame 7 (P-frame) based on frame 6 having a quantizer scale of 5. However, using

[4] Wavelet coding as in MJPEG2000 should also be possible within Evalvid-RA framework.

[5] Still, the different frame types may use different quantizer scales, e.g. I-frames use scale 8 while P-frames use scale 12, and B-frames scale 16, but fixed during the sequence.

[6] This is however not a big sacrifice, since the most challenging research are within interactive media, where the algorithmic delay of CBR should be avoided. However, it also rules out H.264 slice mode: a future Evalvid-RA upgrade to H.264 should therefore include slice granularity trace files as an option.

[7] Here we choose to use the same quantizer scale for all types of frames.

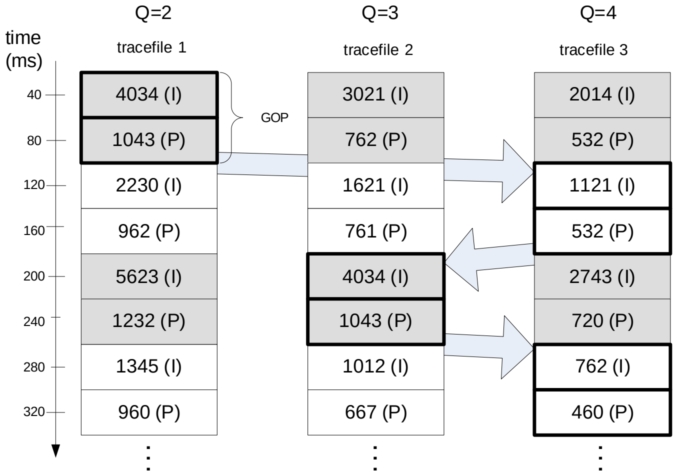

Fig. 1. The Evalvid-RA main concept by letting the simulation time rate controller choose correct frame sizes (emphasized boxes) from distinct trace files valid for each quantizer scale. The figure shows a simplified example of a 25fps video using three quantizer scale values and GOP size of two (one I- and one P-frame).

a separated “offline” encoder and live rate controller (seen from the network simulator), frame 7 is already produced in the pre-process, based on frame 6 having a quantizer scale of 10 also. Although the artifacts produced would not be too much noticeable (verified by own experiments not documented further in this paper), this observation concludes that the only correct option is to have equally sized GOPs and a VBR rate controller that works on GOP granularity.

To summarize this subsection, the pre-process must run an encoder for each media file that shall be used in the network simulation 31 times at open-loop VBR mode (quantizer scale 1–31), and with fixed GOP size (e.g. 12 frames) (see Fig. 2 upper left corner, where the pre-process tools are shown schematically with input and output files). In addition, each of these files must be traced to produce 31 frame size trace files. This process is performed only once for each media file, and the trace files can be used over and over again by new network simulations. The required tools are one encoder and one trace file generator. Since Evalvid-RA 1.0 builds on EvalVid v1.2, the codec choice was limited to MPEG-4 encoders. (However, the current EvalVid 2.0 supports also H.263 and H.264 bit streams. In principle every codec which can be encapsulated in an MP4-container as defined in ISO/IEC 14496-12 and -14 could be used.) In this paper we have used ffmpeg’s MPEG-4 encoder [41] configured to produce equally sized GOPs with fixed quantizer scale. The EvalVid v1.2 `mp4.exe` program has been used to produce the trace files.

C. The network simulation

The next step is the network simulation as shown in Fig. 2 (upper right corner). In a real network, the flows in progress will naturally consist of flows carrying independent and different content. In trace file simulation though, it is common to use the same trace files simultaneously as media input for many or all sending nodes. If the starting *position* inside the trace files is decided randomly and independent for each source, and if the trace files are big enough, this will approximate independent and different sources. In addition, the flows starting *time* in the simulation can also be randomized. This solution will also be used here, but in addition, each source is running independent rate controllers, and these can be pre-set to different target bit rate averages. As they in addition will react on independent network load feedback while running independent rate controllers, it can be concluded that the approximation of independent source modeling is even more valid in our case. Also, in Section VI-D and VI-E we will show that different VBR rate controlled media genres give almost the same traffic characteristics. This is confirmed in earlier work, e.g. [18].

To improve the simulator performance, the trace files are read into memory only once to avoid frequent accesses to external files. In our case we have 31 trace files, with equal number of frame size traces. All files are read into memory during simulation initialization, and organized like a matrix (also stored in `frame_size.dat`), similar to the simple depiction in Fig. 1. Along one of the axis is the frame number count (time), while on the other axis is the quantizer scale 1–31. Through simple indexing, the source nodes can start at a randomized frame number count, while the independent rate controllers (explained in Section V-B) calculate the GOPs quantizer scale that is used as index along the other axis.

Typically, one of the sources is selected as the *primary* flow (Fig. 2, S0–D0 path), i.e. the flow that will be included in the post-process (see Fig. 2 lower part), where the received media will be decoded and perceptual quality like PSNR and MOS values are calculated. This flow must be started at frame number 1, so that the decoded media can be directly compared to the original media, frame by frame. The rest of the flows are used as traffic generators of real rate adaptive media. If desired, more than one flow can be selected as a primary flow, and more than one original media file can be used as source material. In the latter case, using N different original media sources, N matrices of frame sizes must be read into memory in the simulation initialization phase.

To assist the post-process, the quantizer scale used for the primary flow must be logged during simulation time. This information is stored in the senders trace file (`st_be_0` in Fig. 2), together with packet sizes and sending times. Given a simulation time MTU parameter, each frame is typically fragmented into several packets. The packets belonging to the same frame are either submitted back-to-back, or smoothed over one frame interval, eventually smoothed by the TFRC sending buffer (see Section V-C), decided by simulation time parameters. Received packets are logged at receiving nodes (e.g. `rd_be_0` in Fig. 2), storing packet number, time, size, and if missing (detected by received packet numbering not being sequential), tagged as lost. The `frame_size.dat` will together with the other simulation time output files support the received media file binary reassembly and decoding to be performed in the post-process.

D. The post-process

The main post-processing functionality is depicted in the lower left corner of Fig. 2. Using the trace files generated

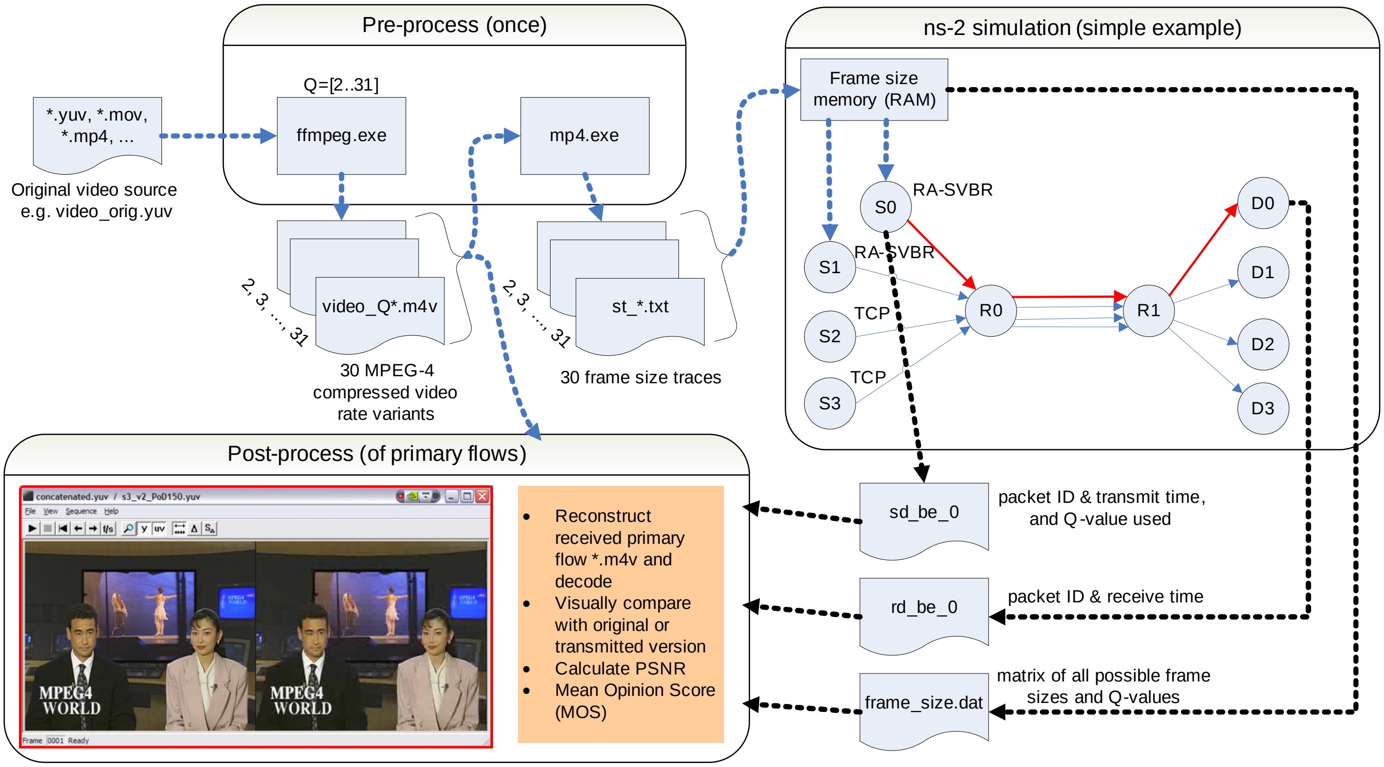
Pre-process (once)
*.yuv, *.mov, *.mp4, ...
Original video source e.g. video_orig.yuv
Q=[2..31]
ffmpeg.exe
mp4.exe
2, 3, ..., 31
video_Q*.m4v
30 MPEG-4 compressed video rate variants
2, 3, ..., 31
st_*.txt
30 frame size traces
ns-2 simulation (simple example)
Frame size memory (RAM)
S0
RA-SVBR
S1
RA-SVBR
S2
TCP
S3
TCP
R0
R1
D0
D1
D2
D3
Post-process (of primary flows)
concatenated.yuv / s3_v2_PoD150.yuv
File View Sequence Help
MPEG4 WORLD
MPEG4 WORLD
Frame 0001 Ready
Reconstruct received primary flow *.m4v and decode
Visually compare with original or transmitted version
Calculate PSNR
Mean Opinion Score (MOS)
sd_be_0
packet ID & transmit time, and Q-value used
rd_be_0
packet ID & receive time
frame_size.dat
matrix of all possible frame sizes and Q-values

Fig. 2. An overview of the Evalvid-RA framework: pre-process, network simulation, and post-process. The 30 trace files st_*.txt serve as input to the network simulator. This example shows two video sources competing for network capacity with two FTP over TCP applications. The source S0 to destination D0 is selected as primary flow.

during network simulation (dashed lines from the right), together with the media files produced during the pre-process (dashed line from the top), several statistics and measures can be calculated from the simulated traffic. As in the original EvalVid [16] with the *ns*-2 interface [17] the following can now be produced:

- loss rate statistics
- delay statistics
- assembly of received compressed media taking packet loss and/or delay into account
- decoding of (possibly) erroneous compressed media
- playing decoded media
- calculate PSNR and/or MOS (decoded media compared to original media)

The first two in the list can be calculated for all flows, while the rest is only available for the primary flow(s). The added functionality, and corresponding challenge, is the assembly of received compressed media. Due to the rate controller, the actual media transmitted is a mix between some or all of the 31 quality variants. Thus, the logging of actual quantizer scale used is a key component, functioning as a pointer to the correct input file. Thus the Evaluate Trace program `et.exe` of EvalVid v1.2 was modified to `et_ra.exe`. It opens all 31 MPEG-4 compressed files for reading, then scans all of them following the size of each compressed GOP and logged quantizer scale, to find the correct start position inside the used MPEG-4 file of every GOP. In this way the correct binary information is copied into the resulting MPEG-4 file, which is the *rate adaptive primary media file* submitted into the network, given the network state feedback at simulation time. Packet losses during simulation will result in corresponding frame loss. The resulting MPEG-4 file will then typically have a varying quantizer scale, but inside each GOP, the quantizer scale is fixed.

A list of the complete Evalvid-RA tools package is given in the Appendix.

V. Adaptive Rate Controller

Having established the framework guidelines, the online rate controller running at simulation time can now be selected. This rate controller will have very limited input information from the encoder. If assuming connection to a live (online) encoder where low delay is of critical concern, there is no a priori information of the visual complexity of the next frame or GOP. The actual number of bytes spent per frame can however easily be monitored, using the information from the input trace files (depicted as `st_*.txt` in Fig. 2). The rate controller constraints are thus target average bit rate, the bit rate variability allowed, plus a possible peak rate limit, all which can be calculated by the rate controller itself.

A. Shaped VBR (SVBR) — A compelling candidate

The two first constraints can efficiently be controlled by a leaky bucket. Leaky buckets in different variants are also commonly used by most offline and online rate controllers. When searching the literature, the Shaped VBR (SVBR) by M. Hamdi et al. [18] is a compelling candidate, since it is of low

TABLE I

LIST OF TERMS USED IN THIS PAPER AND THEIR RESPECTIVE DEFINITIONS

Term	Definition	Units
r	Leaky bucket rate, i.e. the average video rate	bits/GOP
b	Leaky bucket size	bits
$X(k)$	Leaky bucket fullness at time k	bits
$R(k)$	Leaky bucket input during GOP k	bits
$\hat{R}(k)$	Estimate of Leaky bucket input during GOP k	bits
$Q(k)$	static quantizer scale used during GOP k	1–31
$r'(k)$	adaptive Leaky bucket rate used during GOP k	bits/GOP
$b'(k)$	adaptive Leaky bucket size used during GOP k	bits
G	GOP size	frames
r_{new}	current network update of rate	bits/GOP
r_{old}	previous network update of rate	bits/GOP
$\bar{r}$	averaged leaky rate used for TFRC	bits/GOP
r_i^t	partial TFRC rate feedback (number i of N)	bits/GOP
B_i	TFRC sender buffer backlog at feedback i	packets
d_f	decay factor used for forcing sender buffer to drain	
b''	adaptive Leaky bucket size used for TFRC	bits

complexity, and also designed to work on GOP granularity. Their paper stress however that the quantizer scale q producing the average target bit rate r should optimally be known a priori. We found that this requirement could be relaxed without having significant impact on performance, see Subsection VI-B.

The SVBR is using a leaky bucket $LB(r, b)$ where r is the target average bit rate and b is the bucket size (see Table I for paper variables overview). The larger the bucket size, the more rate variability is allowed, producing a more stable quality [42]. The media packets do *not* experience additional delay because the $LB(r, b)$ is used as a virtual buffer, meaning that the packets go straight into the network (or network sending buffer as in TFRC), but is counted in parallel by the $LB(r, b)$. The latter makes it very suitable for interactive communication. The leaky bucket fullness $X(k)$ is calculated at the start of every GOP k as [18]

$$X(k) = \min\{b, (\max\{0, X(k-1) - r\} + R(k-1))\}, \quad (1)$$

where $R(k-1)$ is the actual bits spent during GOP $k-1$. When $X(k)$ is close to zero, the rate control algorithm behaves as in open loop, i.e. with the quantizer scale $Q(k) = q$. When it is close to b, however, it behaves more like CBR, i.e. $R(k)$ is attempted to be close to r. The quantizer scale $Q(k)$ is then calculated as

$$Q(k) = Q(k-1)R(k-1)/\hat{R}(k) \quad (2)$$

assuming that the scene complexity changes slowly from GOP to GOP (i.e. it follows a predefined rate-distortion curve), and $\hat{R}(k)$ is an estimate of the bits to be spent during GOP k. When the scene complexity increases substantially, (2) will calculate too small $Q(k)$, giving too high $R(k)$. This will be "compensated" for in the GOP $k+1$. For pre-stored media and live media allowing a delay equal to one GOP, the next GOP scene complexity will be known a priori, and such bit rate over-shoots can be avoided. For more details on how to calculate $\hat{R}(k)$ we refer to [18].

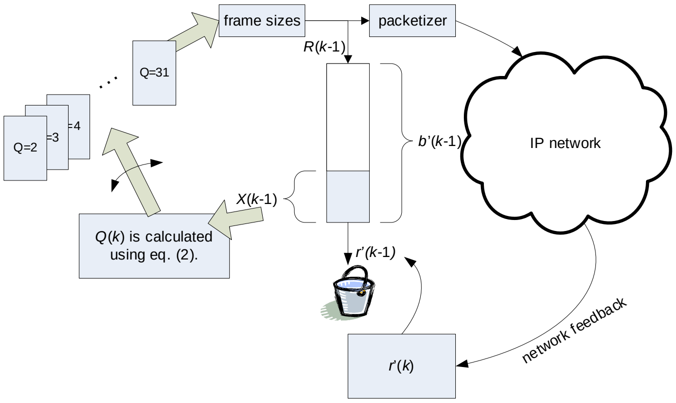

Fig. 3. RA-SVBR with the updates from the network and its selection of frame size information from the available trace files (eventually real frames from online coder in a real implementation).

B. Rate Adaptive SVBR (RA-SVBR): the needed modification

Although SVBR was designed for static values of r, b, and q, we have found that r and b actually can be variables influenced by network state feedback. Using r and b as upper limit values used when the network is in non-congested state, $r'(k) < r$ and $b'(k) < b$ can be calculated whenever the congestion control algorithm suggests a new allowed average bit rate r_{new}. Since these events are *not* synchronized to the GOP periods, (1) must be modified to take this into account. Scaling the bucket size $b' = br'/r$ (the time index k is omitted in the time varying r' and b' from now on), and letting $i \in [0, G-1]$ being the time index for the network feedback event counted as the position in the active GOP of size G frames, the equation becomes

$$X(k) = \min\{b', (\max\{0, X(k-1) - r'\} + R(k-1))\}, \quad (3)$$

where $r' = r_{old}i/G + r_{new}(G-i)/G$ and $b' = b_{old}i/G + b_{new}(G-i)/G$. When there is no network feedback during a GOP, $r' = r_{old} = r_{new}$ and $b' = b_{old} = b_{new}$. We have named this SVBR modification *RA-SVBR*. Fig. 3. depicts an overview over RA-SVBR local operation and its interface towards the live network feedback (right) and media encoder trace files (left).

The major limitation of a GOP based rate controller is that the new rate might be delayed up to the time duration of one GOP before effectuated, depending on the time the network feedback event occurs relative to the local GOP period. The result might be packet drops in the network due to traffic overload. However, (3) makes sure that the bit budget is corrected in the next GOP period. A more complex rate controller could take advantage of the possibility of changing the quantizer scale parameter one or multiple times during a GOP, as discussed in Subsection IV-B.

The major advantage of the rate adaptive version of SVBR is that *any* $r' < r$ can be supported, provided that the quantizer scale needed is within its upper limit. One is not restricted to supporting only the 31 discrete quality variant bit rates — the rate controller ensures that any $r' < r$ can be supported, when averaged over some few number of GOP periods. It is important to note this, since this makes a significant difference to multirate coding where typically only 3–4 different rates are supported.

Very small r' forces the rate controller to select large quantizer scale values. The general video quality when using the very highest quantizer scale values is not very good — visible blocking artifacts show up. In addition, since the quantizer scale value is upwardly bounded to 31, an arbitrary small r' can not be supported. Thus, $r_{min} < r' < r$, where r_{min} is dependent on the current scene complexity.

In practice, this means that also other rate scaling techniques could be considered, such as lowering the frame rate and/or reducing the spatial resolution. Such changes can be supported by signaling repeated headers to the receiver, giving new values for these parameters to the decoder. The simulator implementation could also take lower bounds on quality into consideration and alternatively terminate a session if the allowed throughput is too small. Such information could of course also be used as input to admission control systems in order to prevent starting new flows when available bandwidth is too small. All these features are on the priority list for future Evalvid-RA updates.

C. Supported network feedback systems

In general, any congestion control algorithm can be supported. For best possible stability and link utilization, an average rate limit should be calculated and used as r'. In this paper, two different congestion control mechanisms are tested and compared using the Evalvid-RA tool-set: TFRC [9] and P-AQM+ECF [14], where the latter is a proprietary solution where more accurate network state information is exploited.

The two methods differ significantly. Whereas TFRC relies on either packet drop statistics or ECN tagging (e.g. from RED routers) observed at receiver and signaled back to sender using acknowledgment packets, P-AQM+ECF uses explicit packets with congestion state information based on both input rate and queue statistics directly from each P-AQM enabled router on the path. Furthermore, TFRC requires each packet to be of similar size (TFRC is packet rate oriented, not bit rate — TFRC-SP is another TFRC profile where packets per second is constant and size per packet is a variable, to better suite VoIP applications [43]), while P-AQM+ECF do not impose any such limitations. Since the output from the VBR encoder can not be guaranteed to produce frame sizes that can be fragmented into integer number of packets, byte stuffing has to be used where the actual packet size is less than the fixed TFRC packet size. Clearly, this is bandwidth waste. Even further, TFRC uses strict traffic shaping, in that the TFRC rate is the maximum rate: packets in the transmit queue are submitted at the TFRC packet rate, as long as there are packets in the queue. In P-AQM+ECF, the packets are submitted directly into the network without any traffic shaping. The benefit with the latter approach is no additional transmit buffer delay, while the disadvantage is much more bursty traffic. However, as will be shown by simulations in Section VI-D, there is no significant LRD (Long Range Dependency), so the router buffer occupancy should be controllable.

P-AQM+ECF calculate r' directly, and is interfaced to adaptive SVBR by just passing this value. However, since TFRC is using a transmit buffer, there is a need for a small modification to (3) to ensure that the buffer queue is kept reasonably small. The coupling between the TFRC packet rate and adaptive SVBR is therefore decided to be given as

$$X(k) = \min\{b'', (\max\{0, X(k-1) - \bar{r}\} + R(k-1))\}, \quad (4)$$

where $\bar{r} = 1/N(k-1)\sum_{i=1}^{N} r_i^t e^{-B_i/d_f}$, r^t being the TFRC rate calculated as bytes per GOP and B_i is the instantaneous TFRC transmit queue backlog at the TFRC rate feedback events and $b'' = b\bar{r}/r$. The averaging operation in (4) is necessary due to the fact that TFRC feeds back N updates per GOP. The term e^{-B_i/d_f} with the decay factor $d_f = 100$ ensures that the queue backlog is drained over time. A smaller decay factor than 100 would have drained the queue faster, but we observed that the TFRC feedback system became unstable (queue oscillations bigger and bigger). We also simulated with $d_f = 1000$ to show increased stability at the cost of some increased shaping buffer delay.

VI. Example Evalvid-RA simulation and results

To demonstrate the capabilities of Evalvid-RA some simulation examples are described and the results are discussed in this section. The ns-2 simulation model runs the RA-SVBR source and a dumbbell network topology providing feedback as depicted in Fig. 3. The actual video sources used are described in the next section.

A. Test sequences and the Evalvid-RA pre-process

The video clips for the initial simulations were selected from the official MPEG test clips. This way, our results can be verified by independent researchers. A 1836 frame video sequence was created using a collage of the clips (in given order) News, Football, Akiyo, Stefan, and Paris, at CIF resolution and 30fps (giving a duration of 61.2s). These clips can be downloaded from e.g. http://www.tkn.tu-berlin.de/research/evalvid/cif.html. All sources were using this sequence, however started at different time and frame number (and looped to enable continuous media), thus avoiding traffic synchronization as discussed earlier. The simulation study also covers more elaborate simulations, testing five different IP router architectures. For that study, seven minute long clips from The Matrix (genre “Action movie”, CIF, 29.97 fps) and from “An Inconvenient Truth” (genre “Documentary”, CIF, 25 fps) are used to create even more realistic network traffic. The latter media can also be considered as advanced videoconferencing content, in that there are shots with text, presenter in front of slides with computer graphics, and some shots with natural image content.

Following the Evalvid-RA pre-process these sequences were first compressed with ffmpeg 30 times (static quantizer scale values ranging 2–31 are supported by ffmpeg). The GOP size was fixed to 12 frames with B-frames turned off to avoid algorithmic codec delay[8]. Then the `mp4.exe` trace tool was used on each of these MPEG-4 files to produce the ASCII trace files giving the compressed frame size and type. These 30 trace files were used as basis for the Evalvid-RA *ns*-2 traffic generator `vbr_rateadaptive.cc` in order to produce realistic video traffic, where each frame is fragmented into MTU sized packets before submission. Note that an optimal packetizer would fragment frames to packets at macro block boundaries – similar to the *slices* defined in H.264 – to enhance error resilience. This approach is not possible in the current version of Evalvid-RA since the trace files from the `mp4.exe` tool are generated with frame size granularity, not macro-block or slice granularity.

B. Adaptive SVBR performance vs. ffmpeg's VBR controller

As a first validation of the implementation, a comparison of the RA-SVBR and ffmpeg's rate controller was performed. Using the MPEG test sequence, both RA-SVBR's r-parameter and ffmpeg's own 1-pass VBR rate controller (using the -b switch) were set to 600kbit/s. b in RA-SVBR was set equivalent to 1.5 GOP size in bytes. We noted that ffmpeg's rate controller used some GOP's before stabilizing the rate output, at start it was a bit too high. RA-SVBR was simulated in *ns*-2 using the MPEG sequence produced as described in subsection VI-A. There were no bandwidth bottlenecks and network feedback reading was turned off. This ensured that the RA-SVBR rate controller was working at $r' = r = 600$kbit/s fixed during the whole session. The resulting *ns*-2 trace files were used as input to the `et_ra.exe` tool for MPEG-4 file assembly. This file and the file generated by ffmpeg were now decoded with ffmpeg to raw YUV files. These two YUVs were compared to the original MPEG test sequence to produce the PSNR results, which are shown in Fig. 4a). There is only a minor difference in performance. Not surprisingly, the ffmpeg's own rate controller produces the best result, since it can vary the quantizer step from frame to frame, and even macro-block to macro-block, and not only from GOP to GOP as in RA-SVBR. Inspecting the figure more closely, one can see that the I-frames have significantly better PSNR (about 1.5–2.0 dB), while the P-frames have almost the same PSNR. This is achieved by lowering the quantizer scale of the I-frames, thus producing a better I-frame which is also a better key-frame for the motion estimation of the following P-frames. Nevertheless, this comparison proves that the quantizer scale adjustments made by RA-SVBR and its implementation follow the proposed performance as given in [18].

Fig. 4 a) – c) also show the different complexity of the clips compromising the MPEG test sequence: News (frame 1–300) is medium, football (300–400) high-motion, Akiyo (400–700) is very low complexity, thus the PSNR values get very high, Stefan (tennis player, 700–800) is very complex, thus giving very small PSNR values, and at last Paris (800–1836) which is high to medium.

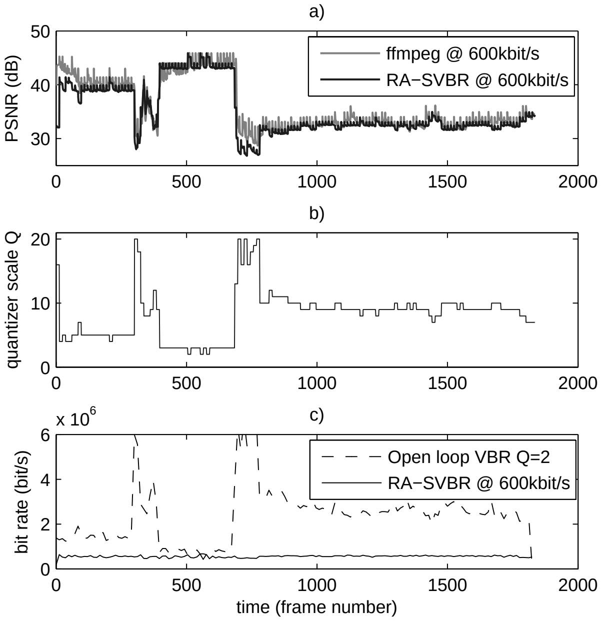

Fig. 4. a) Comparison of PSNR values of RA-SVBR and ffmpeg's rate controller in test sequence. b) The quantizer scale values Q used by RA-SVBR in test sequence in a). c) The bit rate of $Q = 2$ VBR and RA-VBR at 600kbit/s.

C. TFRC and P-AQM initial performance comparison

In this section, a simple scenario with VBR traffic only is tested using the MPEG sequence, in order to address behavior specific for homogeneous video traffic and network characteristics. The TFRC streaming media flows are routed through a network with either ordinary FIFO or RED routers with ECN enabled, while adaptive UDP is streamed over P-AQM with ECF signaling. A simple dumbbell network topology was used. The bottleneck link capacity was 40 Mbit/s with a propagation delay of 10 ms. The access network capacities were 32 Mbit/s[9] with 5 ms delay (each side of the bottleneck link), thus producing a total one-way propagation delay of 20 ms. 64 media sources were started at random time, uniformly distributed over the first 16 s period of simulation time, but all ended simultaneously at 64 s. The only exception was the primary flow that started at 10ms and ended at 61.21 s. Each source had a target RA-SVBR average bit rate set to $r = 1.0$ Mbit/s. The fair share bandwidth after all sources have started was however 40 Mbit/s$/64 = 625$ kbit/s. The challenge of the network congestion control and the rate adaptive SVBR was then to make the sources produce 625 kbit/s on average (after 16 s, packet headers included), ensuring bandwidth fairness and smallest possible delay between sender and receiver. The end-to-end delay budget includes sender buffer (TFRC only), packet transmission delay, propagation delay, and network router queuing delay. MTU was set to 1036 bytes for the TFRC case, and 1028 bytes for the P-AQM case. These numbers resulted from 1000 byte payload, 20 byte IP header, 8 byte UDP header (P-AQM) and 16 byte DCCP/TFRC header

[8] B-frames are however fully supported by Evalvid-RA.

[9] to make sure that the access network does not cause any form of queuing

TABLE II
NS-2 SIMULATION RESULTS

Sim. #	Cong. Control	d_f	Utiliz. (%)	P. drop (%)
s1	P-AQM+ECF	—	88.2	0.0
s2	RED/ECN+TFRC	1000	89.9	0.0
s3	RED/ECN+TFRC	100	90.0	0.0
s4	RED/ECN+TFRC	40	89.6	0.0
s5	FIFO+TFRC	100	92.0	1.1

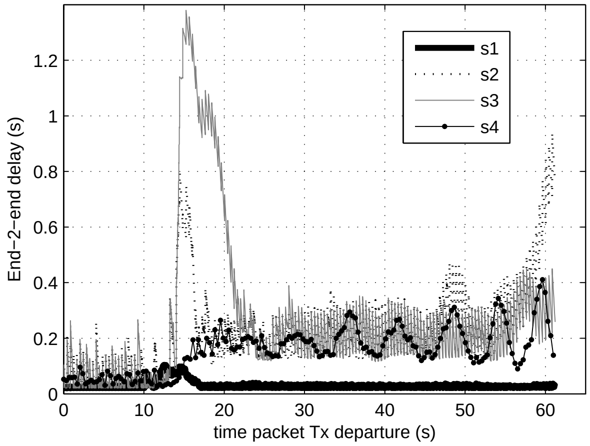

Fig. 5. The packet delay end-2-end for the primary flow, including traffic shaping buffer, transmit delay, propagation delay and router queue delay.

[8]. In a real implementation the RTP protocol could have been used additionally — this would have added typically 12 bytes. The RED router (used by the TFRC simulations) was configured to gentle adaptive RED with the target delay set to half of the maximum queue buffer size. The buffer size was set equal to the bandwidth-delay product (BDP) assuming an RTT of 200 ms[10], which gives 0.200 x 40e6 / 8 = 1 MB, i.e. approximately 1000 packets (assuming 1000 byte packets). The RED target queue equilibrium was thus about 500 packets. Smaller queue equilibrium was also tried but resulted in severe queue length instability. P-AQM, which is designed to control aggregate traffic with small persistent queue sizes, was configured to a target queue size of only 50 kB. Both RED and P-AQM were run in byte count mode. Transmitter (encoder) frame discard as additional rate control was not allowed.

Table II lists the simulations and their parameters and results, showing $\sim 90\%$ link utilization and zero loss for all the simulation cases, except s5 which uses packet drops to signal congestion.

Fig. 5 shows the end-to-end delay for simulation cases s1–s4. S1 (P-AQM) has very low delay, at equilibrium it is below 30ms. The TFRC simulations s2–s3 show that there is a significant period in which the delay is very high. An inspection shows that this delay is both due to excessive queue delay and significant shaping buffer backlog. S4 shows that a too low decay factor leads to unstable behavior. We believe, the reason is that the stable packet submission of TFRC is discontinued by the completely drained shaping buffer. The TFRC "Fast Restart" functionality, which should assist in stability for self-limiting sources, was however enabled.

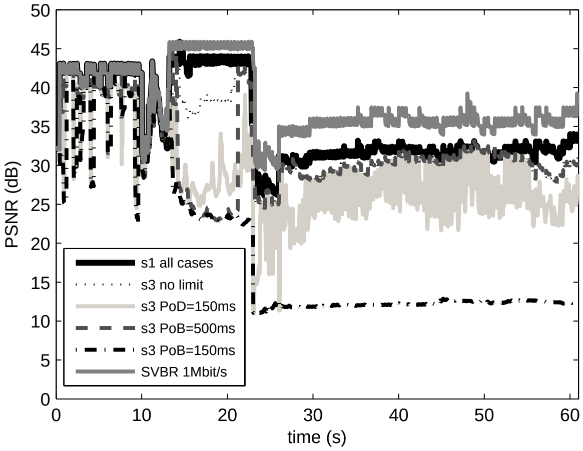

Fig. 6. The resulting PSNR values (frame by frame) of the primary flows in s1 and s3 simulation, given the different delay constraints.

The Evalvid-RA post-processing tools for the primary flows were now used to generate PSNR and MOS values for s1–s3, given three different delay constraint scenarios: (i) no delay constraint, (ii) receiver play-out buffer size constraint (PoB), and (iii) absolute play-out buffer time constraint (PoD). In (i) all received packets were used in the frame assembly process (by `et_ra.exe`), while in (ii) packets were dropped if the packet inter-arrival jitter was higher than a specified receiver play-out buffer size could tolerate (due to memory limitation). In (iii), an absolute play-out time was specified relative to the frame transmission time, due to the real-time constraint. We tested the simulated scenario with 150 ms and 500 ms equivalent play-out buffer size constraint and 150 ms absolute play-out delay constraint. The latter reflecting the recommended one-way delay for conversational media. Fig. 6 displays the results for s1 and s3. Since these scenarios had zero loss (due to the ECN and ECF signaling), the PSNR values reflect two other QoS parameters: bandwidth and delay. Bandwidth fairness can be examined by calculating per flow bandwidth, using e.g. Jain's Fairness Index [45]: it was better than 0.99 for all simulations (1.0 is perfect fairness), showing that the bandwidth was fairly distributed over the flows. P-AQM performs best at all tests, due to its superior end-to-end delay performance, both in receive frame jitter and absolute delay. However, the delay caused by TFRCs traffic shaping buffer and RED router affected the perceived quality of TFRC in terms of objective PSNR values. Constraint (i) gives almost the same PSNR as P-AQM, while (iii) shows that the absolute delay of 150ms results in PSNR degradation in the order 1–10dB. This degradation is due to the fact that the decoder has to render the last successfully decoded frame when the current frame number is not yet arrived at the receiver. In case (ii), a PoB of 500 ms is sufficient to handle most of the inter-arrival

[10] The RED router must be set to cope with typical average RTT of the flows traversing it, and not the special case with low RTTs as in this example; this also makes it more robust to handle many flows, see e.g. [44].

TABLE III

EVALVID-RA POST-PROCESSING RESULTS

Sim. #	avr. delay (ms)	max delay (ms)	avr. PSNR (dB)				frames slipped / total fr.
			PoB $=\infty$	PoB= 500ms	PoB= 150ms	PoD= 150ms	
s1	30.3	104	35.0	35.0	35.0	35.0	0/1836
s2	212.1	939.4	33.7	32.8	19.1	29.2	1259/1836
s3	231.2	1379	33.1	31.5	19.3	28.8	1250/1836

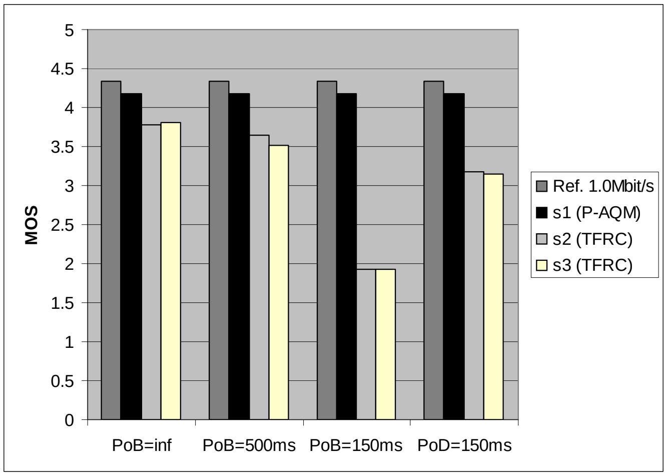

Fig. 7. Average MOS values calculated from the PSNR values following guidelines in [16], [39]. A reference MOS value is calculated for a 1.0 Mbit/s flow of the same sequence, which would have resulted if there were fewer than 40 flows in the bottleneck.

packet jitter to avoid too much PSNR degradation, while with a PoB of 150 ms, a lot of frames will be dropped due to buffer limitation so that decoding collapses. Statistical delay and PSNR values for the tests s1–s3 are shown in Table III, with corresponding average MOS values shown in Fig. 7.

This subsection has demonstrated that the perceptual quality of interactive video flows is not only a function of bandwidth and packet drop ratio, but also on end-to-end delay. The network feedback systems are shown to cooperate closely with the adaptive rate controller so that the aggregate traffic gives link utilization close to capacity while packet drops are limited. Due to the inherent TFRC traffic shaping, it is probably natural that this non-bursty traffic can be strictly controlled. The non-traffic shaped traffic output of the P-AQM+ECF system is however not so evident since it submits the VBR traffic directly into the network. The reason why this works well is examined in the next subsection.

D. Adaptive VBR rate control avoiding LRD

In [20] and also later work by others it was proven and demonstrated that VBR video traffic exhibits long range dependence (LRD). LRD traffic characteristic means that the resulting rate (measured in bytes per frame or per GOP) occupied by the VBR coder varies significantly and that its ACF (autocorrelation function) has significant values for large lags n, i.e. the ACF $\rho(n) \propto n^{-\beta}$ as $n \to \infty$ and $0 < \beta < 1$ (compared to the exponential fast decay $\rho(n) \propto \alpha^n$, $n \to \infty$ and $0 < \alpha < 1$, valid for Poisson sources). With other words, the VBR coder traffic output has a self-similar behavior.

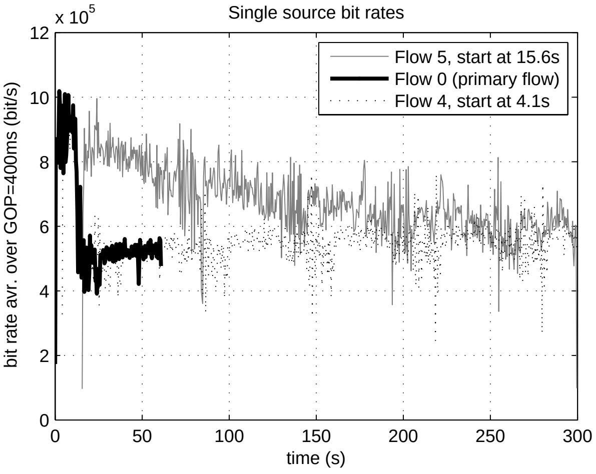

Fig. 8. Three of the 64 flows, showing the VBR behavior, and the adaptive rate control slowly adjusting the rate to the 600kbit/s fair application rate.

Obviously, such traffic makes it very difficult to have high link utilization without risking periods with persistent packet losses due to queue buffer overflow. However, the work cited did statistical analysis of *VBR open-loop* coders only, i.e. no rate controller was present at all. Applying VBR rate control means that an average bit rate is established, possibly also with variance constraints. This is exactly what is gained by adaptive SVBR in the form of (3) and (4). [18] also shows that the rate controller almost completely eliminates any LRD, i.e. it becomes more like SRD (short range dependent). *This is why the deployment of VBR rate controllers makes high link utilization obtainable, since the aggregate of SRD sources will exhibit Poisson characteristics.* When scaling both the r and b of the leaky bucket in SVBR, variability is also reduced per source to adjust to the potentially increased variance of the aggregate. Thus, congestion control combined with *adaptive* rate controllers makes way for even more flows and stabilizes the network throughput at high utilization. The accuracy of the feedback system and buffer dimensioning then determines if this can be accomplished with small buffer delays.

An Evalvid-RA ns-2 simulation was carried out to substantiate the claims made above. It was similar to the P-AQM simulation described in the previous subsection, except that it was run over 300 seconds to get more data for the statistics. All flows were looped back to the beginning of the trace files when finished, except for the primary flow that stopped at 61.2 s as before. In Fig. 8 the primary flow rate is shown together with flow 4 and 5. Note that since flow 5 starts at 15.6 s, it is one of the last flows to start in the 0–16 s starting period, thus its convergence against the fair bandwidth share is slower than “normal” (e.g. compared to flow 4). This plot shows that it takes some time before the flows become stationary. However, the *aggregate* of the flows entering the bottleneck router has a stationary behavior much sooner, as shown in Fig. 9. The reason for this is that the congestion control of P-AQM works on the aggregate, while the AIMD behavior of the sources themselves control the fairness issue. Here the aggregate bit rate has been calculated using four

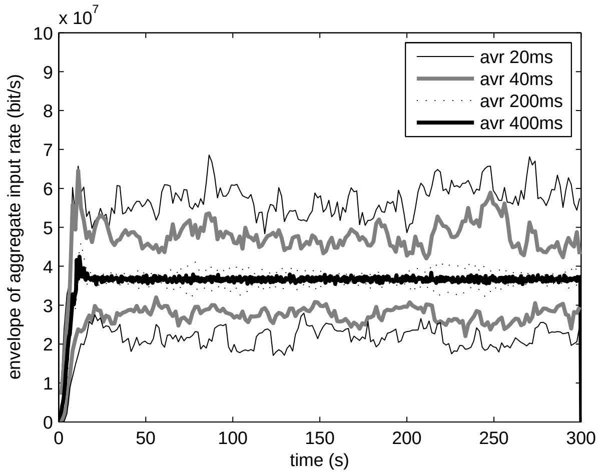

Fig. 9. Averaging at larger and larger time scales reveals a stationary time series.

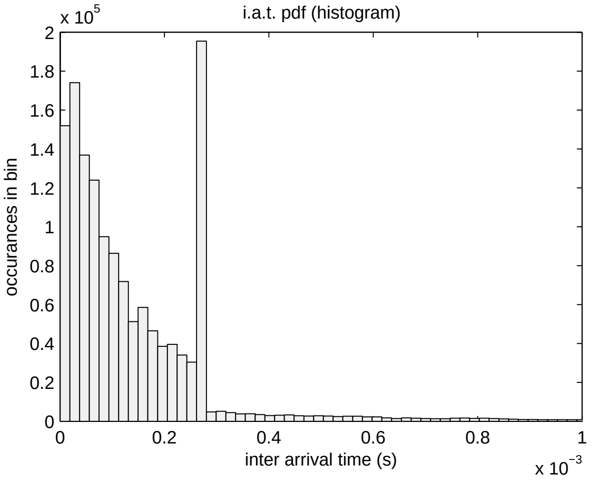

Fig. 10. The histogram of the inter arrival time of packet received at bottleneck router

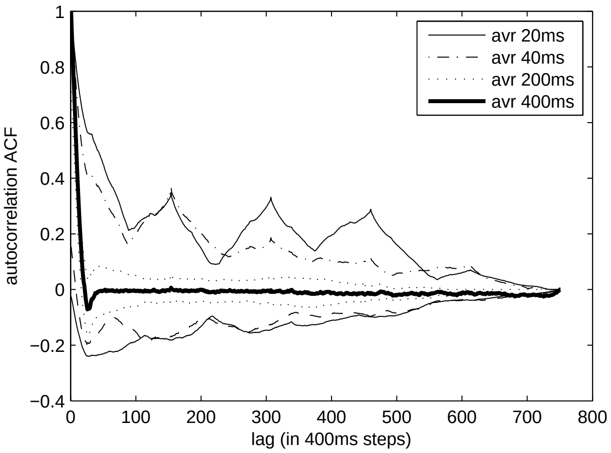

Fig. 11. The envelope of the autocorrelation function of aggregate input traffic to bottleneck router, calculated at four different time units. Lag units are scaled to fit corresponding time unit.

different averaging time units: 20, 40, 200, and 400 ms (=GOP). As shown by the curve for GOP sized averaging rates, stationary behavior is obtained already at approximately 20 s. Its variability at smaller time scales is much higher. However, the figure shows that the averaging operation reduces the variance considerably, which is typical for Poisson and Poisson-like traffic aggregates. Calculating a histogram of the packet inter arrival times (Fig. 10) reveals that the traffic is indeed Poisson-like, since a negative exponential distribution shape is produced. The only exception is the spike at 0.27 ms, caused by the frequent event of multi-packet frames arriving back-to-back (1028 B packets in 32 Mbit/s access links have 0.27 ms spacing). Calculating the autocorrelation function of the bit rates as shown in Fig. 9, gives the results as shown in Fig. 11. With 400 ms average time windows, the sequence is clearly uncorrelated. At smaller time scales a correlation peak at the lag corresponding to 61.2 s is evident. This is not surprising as the flows repeat themselves after this amount of time. This is a result of a “synthetic” aggregate behavior and motivates a modification of Evalvid-RA to jump to an arbitrary GOP after ending the trace file instead of jumping to the very beginning. Nevertheless, the envelope shape reveals that the ACF converges fast to zero at increasing lag, as is the nature of Poisson-like traffic sources.

It is the near Poisson traffic nature that makes it possible to control an aggregate of VBR rate controlled video streams close to full link utilization, with zero packet loss and very small queue delay. As future bottleneck router capacity increases, higher link utilization is obtainable without adding delay, possibly even decreasing the delay at the same link utilization.

E. Mixed VBR and TCP traffic

In this section we aim to demonstrate the Evalvid-RA capabilities in video transmission protocol analysis using more realistic Internet traffic and running a high number of different work loads in order to compare the different protocols and network architectures. The focus is on relative performance, thus we present the results as PSNR values as function of the number of VBR flows.

A common bottleneck link of 40 Mbit/s is shared by 32 long-lived New Reno TCP flows (e.g. continuous FTP download) and 120 sources generating HTTP Web traffic using a recommended model generating Poisson distributed flow arrival times and Pareto distributed flow sizes (with shape factor of 1.35) [46]. The access network capacity is 3.0 Mbit/s, while the rest of the parameters are similar to Section VI-C. In this environment the VBR flows are transmitted. We vary the number of VBR flows from 2 up to 128, using the clips from ”The Matrix” and ”An Inconvenient Truth”. In addition to P-AQM and TFRC over RED/ECN routers (TFRC 1), we also test TFRC over RED without ECN marking (TFRC 2), TFRC over ordinary FIFO routers (TFRC 3), and non-adaptive 1.0 Mbit/s UDP flows over FIFO routers (UDP). To obtain reference PSNR values (”ref.” curves in Fig 12 and 13), we also simulated a single UDP flow with target bit rates 1.0 Mbit/s, 740 kbit/s, 570 kbit/s, 392 kbit/s, and

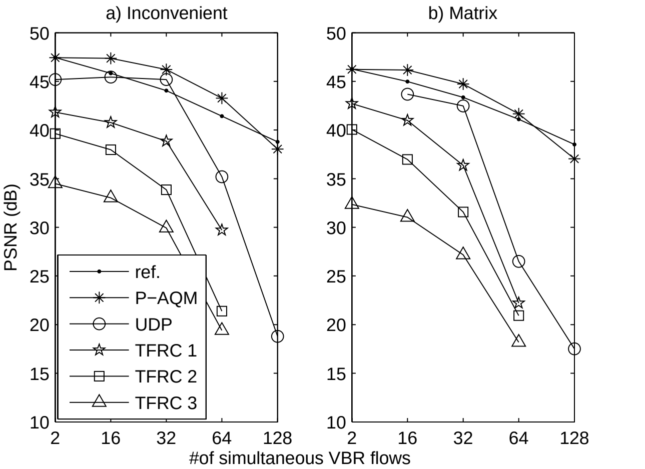

Fig. 12. PSNR values as function of number of VBR flows in mixed network traffic. Play-out delay constraint is 150ms (videoconferencing delay constraint).

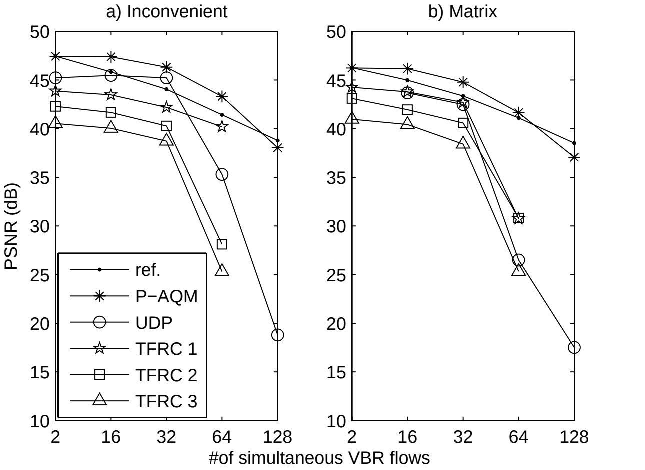

Fig. 13. PSNR values as function of number of VBR flows in mixed network traffic. Play-out delay constraint is 2s (VoD and WebTV delay constraint).

240 kbit/s, corresponding to the fair bandwidth share in the different cases.

In Fig. 12 the results for the videoconferencing 150 ms delay constraint case are depicted, while in Fig. 13 the corresponding results for the VoD/WebTV 2 s delay constraint case are shown. The P-AQM performance is equal and above the reference quality, where the reason for the latter is in fact *statistical multiplexing gain* (SMG): since both movie clips have large variations in their bit rate for all quantizer scales Q, and for $Q = 2$ the bit rate range is approx. 0.25–4.5 Mbit/s, there is room for other flows to exploit other flows inability to fully utilize their fair bandwidth share. The flows are upper bit rate limited at 1.0 Mbit/s, explaining the absence of SMG at 2 VBR flows. With 128 VBR flows, the fair bandwidth share is below the minimum bandwidth at $Q = 2$, which also renders SMG impossible. P-AQM is robust also in mixed traffic due to a two-queue scheduler that separates the UDP and TCP traffic and marks the TCP packets with ECN as in RED routers [14]. Non-adaptive UDP streaming also achieves very high PSNR values (but not has high as P-AQM due to packet losses) before it collapses above 32 flows due to very high packet losses. It must be noted that the high performance of non-adaptive UDP is on the cost of starved TCP flows! It is also evident that TFRC performs best when run over ECN enabled RED routers. Performance drops a little when ECN is not supported, while ordinary FIFO queues gives TFRC the lowest performance. The better quality of TFRC at the 2 s delay constraint is due to the fact that most frames do arrive with a latency between 150 ms and 2 s. Again, like in Section VI-C, this delay is a combination of traffic shaping buffer delay and RED and FIFO queuing delay. TFRC also pays a PSNR penalty at any number of flows, in that it has constant packet size, and thus must often use bandwidth-wasting byte stuffing.

All simulated cases had bottleneck link utilization above 99.0%. Packet drops increased with the number of VBR flows: for P-AQM it was in the range 0.001–1%, for TFRC 0.01% with ECN, 0.1–2% without ECN and with FIFO, and ill-behaving UDP 0.6–89%. Jain's Fairness Index of the VBR flows was better than 0.99 for all TFRC and P-AQM simulations. When comparing all long-lived flows, the index was 0.96 or better. Also worth noting is that the results for the two media clips were very similar, demonstrating that VBR rate control reduces LRD and thereby genre differences.

VII. Closing Remarks and Conclusion

In this paper we have presented Evalvid-RA, a framework and tool-set to enable simulation of rate adaptive VBR video. Evalvid-RA's main capability is the generation of true rate adaptive MPEG-4 VBR traffic, i.e. the codec output is dependent of the aggregate traffic passing through the network bottlenecks. In addition, the received media traces are used to restore true media files that can be visually inspected and PSNR and MOS scores can be calculated when comparing with the original material. The tool-set includes an online (at simulation time) rate controller that, based on network congestion signals, chooses video quality and bit rates from corresponding pre-processed trace files.

Evalvid-RA's capabilities were demonstrated by the simulation of a VBR rate controller, modulated by TFRC and P-AQM+ECF congestion signals. Up to 128 simultaneous independent VBR sources were run, together with 32 long-lived TCP flows and background Web traffic generated by 120 independent sources. The 420 second network simulation took about 10 minutes to complete on a three year old laptop running ns-2 under Cygwin. Thus, even higher numbers of sources should be feasible.

Statistical analysis of P-AQM+ECF controlled VBR traffic revealed that the traffic aggregate did not exhibit self-similarity. That's why high link utilization and controlled queuing delay and packet loss is obtainable without strict traffic shaping as e.g. TFRC is using. The P-AQM system had both the highest PSNR score and could also support more flows at reasonably high PSNR values. The cost of these achievements is however a new router algorithm (at least located at the bottleneck link) and some additional signaling traffic. The corresponding simulations of TFRC revealed that the performance was increasing with higher network router intelligence. They also showed that delay constraint results were both dependent on the traffic shaping buffer backlog and router queue backlog. Our solution of draining the traffic shaping buffer could probably be improved, e.g. by using frame discard if the buffer contains more than e.g. 2–3 frames, depending on the application. But then, also other means should be developed that prevent unstable TFRC behavior when using such aggressive buffer draining.

Evalvid-RA can be used as a test tool for new ideas and early implementations. The usage of TFRC for media applications is expected to grow substantially in the coming years and improved performance in real-time applications with strict delay constraint, such as videoconferencing, would make it even more valuable. Obviously, the Internet community prefers simple scalable solutions over new ideas involving e.g. new router architecture as in P-AQM. However, this does not prevent the use of novel architectures in dedicated media networks, such as digital-TV.

More advanced Evalvid-RA usage includes fairness and delay performance tests in scenarios with multiple bottlenecks, heterogeneous RTTs, and scenarios where some sources are self-limited while others are bottleneck limited [47]. Advanced routers with selective packet drop can be tested with new error resilient media features, since PSNR and MOS scores can be calculated in the Evalvid-RA post process. Work on rate adaptive media over wireless networks will be more and more relevant. In fact, such work has already been started at NTNU using Evalvid-RA and ns-2 802.11 models.

Future tool enhancements could include support for audio codecs and more video codecs (such as H.264/AVC, which is already supported by Evalvid 2.0), as well as transmitter frame discard and relaxed quantizer scale constraints. The quantizer scale modulation demonstrated in this paper can in fact be expanded to also include temporal and spatial scalability, perhaps even modality changes, provided that the scaling follows a predefined rate-distortion curve. Ordinary multirate coding can be supported, with trace files resulting from optimized CBR or VBR rate controlled media, with arbitrary quantizer scale values on frame, slice, or even macro block granularity. In fact, this awakens the idea of using the multiple precoded media with fixed quantizer scale (as used in Evalvid-RA to simulate real-time codecs) also as content on real streaming servers, thus enabling streaming media services of pre-stored VoD content with rate adaptation at much finer granularity than ordinary multirate coding. Some sample tests reveal that the additional storage cost is six times that of storing only the highest quality stream, which can be justified with the dropping prices of storage media. In such a way Evalvid-RA could also become not only an analysis concept, but also a concept of implementation, and a bridge in rate adaptive media deployment.

By publishing the Evalvid-RA source code online, we hope that the Internet real-time media research community successfully uses this tool-set to investigate, develop, and optimize adaptive media codecs and network architecture jointly, so that current and future adaptive packet video systems are better suited to handle the varying wired and wireless network capabilities and conditions. The latest version of Evalvid-RA can be downloaded from `http://www.item.ntnu.no/~arnelie/Evalvid-RA.htm`.

TABLE IV
THE EVALVID-RA TOOLS OVERVIEW: PRE-PROCESS, SIMULATION, AND POST-PROCESS.

Tool	Original Evalvid-RA?	Evalvid-RA script	Purpose
`ffmpeg`	No	`manyQ.sh`	To encode video file with the full range of quantizer scale values 2–31.
`mp4.exe`	No (Evalvid 1.2)	`manyQ.sh`	Create frame size trace files of all encoded files from previous step.
ns-2: `vbr_rateadapt.cc`	Yes	`concat_TFRC*.tcl`	Simulation: Module running RA-SVR and interfacing the frame size trace files and network feedback.
ns-2: `ra_evalvid_udp.{cc,h}`	Yes (i.e. modification of [17])	`concat_TFRC*.tcl`	Simulation: modified udp.cc in where sender trace files are written, including tx time, packet type and Q-value used.
ns-2: `ra_evalvid_udpsink2.{cc,h}`	Yes (i.e. modification of [17])	`concat_TFRC*.tcl`	Simulation: modified udpsink.cc in where receiver trace files are written, including rx time and packet type.
ns-2: awk scripts	Yes	See `commands.txt`	Sample scripts for simple post-processing of ordinary ns-2 packet trace files.
`et_ra.exe`	Yes (mod. `et.exe` Evalvid 1.2)	`runPoD.sh` and `runPoB.sh`	Post-process: Re-assembly of the rate adaptive MPEG-4 file sent during simulation time.
`fixyuv_ra.exe`	Yes (mod. `fixyuv.exe` Eval. 1.2)	`runPoD.sh`	Post-process: Inserts missing frames due to drop or late arrival so that sent and received video consists of equal number of frames.
`psnr.exe`	No (Evalvid 1.2)	`runPoD.sh` and `runPoB.sh`	Post-process: Calculate the PSNR.
`mos.exe`	No (Evalvid 1.2)	`runPoD.sh` and `runPoB.sh`	Post-process: Map MOS values from PSNR.
`miv.exe`	No (Evalvid 1.2)	`runPoD.sh` and `runPoB.sh`	Post-process: Calculate quality indicator for longer sequences.

APPENDIX
LISTING OF THE EVALVID-RA TOOLS

Table IV is included to ease the understanding of what tools are included in the Evalvid-RA download package, their origin, their purpose, and how to use them. Since all tools are command-line based, they are accompanied by sample script files (Linux shell scripts and ns-2 TCL scripts).

ACKNOWLEDGMENT

The authors would like to thank Chih-Heng Ke (NCKU Taiwan) who wrote the original ns-2 interface for the (non-rate adaptive) Evalvid. We also would like to thank The Research Council of Norway for the support of Mr. Lie's Ph.D. work, and the people at DResearch Digital Media Systems for supporting Mr. Klaue's research.

REFERENCES

[1] P. A. Palumbo, “Broadband streaming video: Viewer metrics and market growth analysis 2000 - 2004,” Accustream Research, Tech. Rep., 2004.

[2] UNINETT, “Digital Brytningstid Uninett 10år,” UNINETT, Tech. Rep., October 2003. [Online]. Available: http://www.uninett.no/publikasjoner/digital.brytningstid/digital.brytningstid.pdf

[3] “ISO/IEC 13818-2, Information technology – Generic coding of moving pictures and associated audio information – Part 2: Visual,” 1994.

[4] “ISO/IEC 14496-2, Information technology – Coding of audio-visual objects – Part 2: Visual,” 1999.

[5] W. Li, “Overview of fine granularity scalability in MPEG-4 video standard,” *IEEE Trans. Cct. Syst. for Video Tech.*, vol. 11:3, pp. 301–317, March 2001.

[6] S. Wenger, Y.-K. Wang, and M. M. Hannuksela, “RTP payload format for H.264/SVC scalable video coding,” *Journal of Zhejiang University*, vol. 7, no. 5, pp. 657–667, May 2006.

[7] S. Floyd and K. Fall, “Promoting the use of end-to-end congestion control in the Internet,” *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, 1999.

[8] E. Kohler, M. Handley, and S. Floyd, “Datagram Congestion Control Protocol (DCCP),” IETF RFC4340, Tech. Rep., Mar. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4340.txt

[9] S. Floyd, E. Kohler, and J. Padhye, “Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC),” IETF RFC4342, Tech. Rep., Mar. 2006. [Online]. Available: http://www.ietf.org/rfc/rfc4342.txt

[10] T. V. Lakshman, P. P. Mishra, and K. K. Ramakrishnan, “Transporting Compressed Video Over ATM Networks with Explicit Rate Feedback Control,” in *Proceedings of the INFOCOM’97*. Washington, DC, USA: IEEE Computer Society, 1997, p. 38.

[11] R. Rejaie, M. Handley, and D. Estrin, “RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet,” in *Proc. of IEEE Infocom*, March 1999.

[12] M. Miyabayashi, N. Wakamiya, M. Murata, and H. Miyahara, “MPEG-TFRCP: Video Transfer with TCP-friendly Rate Control Protocol,” in *Proc. of IEEE International Conference on Communications (ICC2001)*, vol. 1, June 2001, pp. 137–141.

[13] D. Sisalem and A. Wolisz, “LDA+ TCP-Friendly Adaptation: A Measurement and Comparison Study,” in *Proc. of NOSSDAV*, 2000.

[14] A. Lie, O. M. Aamo, and L. A. Rønningen, “A Performance Comparison Study of DCCP and a Method with non-binary Congestion Metrics for Streaming Media Rate Control,” in *Proc. of 19th International Teletraffic Congress (ITC’19)*, Beijing, China, Aug–Sept 2005.

[15] K. Ramakrishnan, S. Floyd, and D. Black, “The Addition of Explicit Congestion Notification (ECN) to IP,” IETF RFC3168, Tech. Rep., September 2001.

[16] J. Klaue, B. Rathke, and A. Wolisz, “EvalVid - A Framework for Video Transmission and Quality Evaluation,” in *Proc. of the 13th International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, Urbana, Illinois, USA, Sept. 2003.

[17] C.-H. Ke, “How to evaluate MPEG video transmission using the NS2 simulator,” 2004. [Online]. Available: http://hpds.ee.ncku.edu.tw/~smallko/ns2/Evalvid_in_NS2.htm

[18] M. Hamdi, J. W. Roberts, and P. Rolin, “Rate control for VBR video coders in broad-band networks,” *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 6, August 1997.

[19] A. R. Reibman and B. G. Haskell, “Constraints on Variable Bit-Rate Video for ATM Networks,” *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 2, no. 4, pp. 361–372, Dec. 1992.

[20] M. Garrett and W. Willinger, “Analysis, Modeling and Generation of Self-Similar VBR Video Traffic,” in *Proc. of ACM Sigcomm*, London, 1994.

[21] J. Beran, R. Sherman, M. Taqqu, and W. Willinger, “Long-range dependence in variable-bit-rate video traffic,” *IEEE Transactions on Communications*, vol. 43, no. 234, pp. 1566–1579, Feb/Mar/Apr 1995.

[22] M. Krunz and S. K. Tripathi, “On the Characterization of VBR MPEG Streams,” in *Proceedings of ACM Sigmetrics’97*. Seatle, Washington: ACM, May 1997.

[23] N. Ansari, H. Liu, and Y. Q. Shi, “On Modeling MPEG Video Traffics,” *IEEE Trans. on Broadcasting*, vol. 48, December 2002.

[24] C. H. Liew, C. Kodikara, and A. M. Kondoz, “Modelling of MPEG-4 Encoded VBR Video Traffic,” *IEE Electronic Letters*, vol. 40, no. 5, March 2004.

[25] J. Zhu, A. Matrawy, and I. Lambadaris, “Models and tools for simulation of video transmission on wireless networks,” in *Proc. of IEEE Electrical and Computer Engineering*, 2004.

[26] W. Mohsin and M. Siddiqi, “Scalable Video Transmission and Congestion Control using RTP,” Department of Electrical Engineering, Stanford University, Tech. Rep., May 2002.

[27] C. Xu, J. Liu, and C. Zhao, “Performance analysis of transmitting H.263 over DCCP,” in *IEEE Int. Workshop VLSI Design and Video Technology*, May 2005.

[28] L. Xu and J. Helzer, “Media Streaming via TFRC: An Analytical Study of the Impact of TFRC on User-Perceived Media Quality,” in *Proc. of Infocom*, March 2006.

[29] E. Gürses, “Optimal Streaming of Rate Adaptable Video,” Ph.D. dissertation, The Graduate School Of Natural And Applied Sciences Of Middle East Technical University, 2006.

[30] H. V. Balan, L. Eggert, S. Niccolini, and M. Brunner, “An Experimental Evaluation of Voice Qualit over the Datagram Congestion Control protocol,” NEC Europe, Germany, Tech. Rep., 2006.

[31] S. Wolf and M. Pinson, “Video quality measurement techniques,” U.S. Department of Commerce, NTIA, Tech. Rep. 02-392, June 2002.

[32] S. Winkler, *Digital Video Quality – Vision Models and Metrics*. John Wiley & Sons, 2005.

[33] T1.801.03, “Digital transport of one-way video signals – parameters for objective performance assessment,” ANSI, Tech. Rep., 2003.

[34] M. Zink, O. Künzel, J. Schmitt, and R. Steinmetz, “Subjective impression of variations in layer encoded videos,” in *Proceedings of the 11th IEEE/IFIP International Workshop on Quality of Service (IWQoS’03), Monterey, CA, USA*, June 2003, pp. 134–154.

[35] J. Gross, J. Klaue, H. Karl, and A. Wolisz, “Cross-layer optimization of OFDM transmission systems for MPEG-4 video streaming,” *Computer Communications*, vol. 27, pp. 1044–1055, 2004.

[36] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, “Image quality assessment: from error visibility to structural similarity,” *Image Processing, IEEE Transactions on*, vol. 13, no. 4, pp. 600–612, 2004.

[37] “VQM Software.” [Online]. Available: http://www.its.bldrdoc.gov/n3/video/vqmsoftware.htm

[38] Sarnoff, “JNDmetrix.” [Online]. Available: http://www.sarnoff.com/products_services/video_vision/jndmetrix/

[39] J.-R. Ohm, *Digitale Bildcodierung - Repräsentation, Kompression und Übertragung von Bildsignalen*. Springer, 1995.

[40] I. JTC1/SC29/WG11, “Information technology – Coding of audio-visual objects – Part 2: Visual,” 1999, ISO/IEC 14496-2.

[41] LGPL, “FFMPEG Multimedia System.” [Online]. Available: http://ffmpeg.mplayerhq.hu/

[42] T. Lakshman, A. Ortega, and A. Reibman, “VBR Video: Trade-offs and potentials,” *Proceedings of the IEEE*, vol. 86, no. 5, pp. 952–973, May 1998.

[43] S. Floyd and E. Kohler, “TCP Friendly rate Control (TFRC): The Small-Packet (SP) Variant,” IETF RFC4828, Tech. Rep., Apr. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4828.txt

[44] J. Chung and M. Claypool, “Analysis of active queue management ,” in *Second IEEE International Symposium on Network Computing and Applications*, April 2003, pp. 359–366.

[45] R. Jain, D. Chiu, and W. Hawe, “A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Systems,” DEC Research Report TR-301, Tech. Rep., Sept 1984.

[46] D. Katabi, M. Handley, and C. Rohrs, “Congestion Control for High Bandwidth-Delay product Networks,” in *Proc. of ACM Sigcomm*, 2002.

[47] T. Phelan, “TFRC with Self-Limiting Sources,” Sonus Networks, Tech. Rep., Oct 2004. [Online]. Available: http://www.phelan-4.com/dccp/tfrc-self-limit.pdf