

Advances in Channel-adaptive Video Streaming

Bernd Girod, Mark Kalman, Yi J. Liang, and Rui Zhang

Information Systems Laboratory

Department of Electrical Engineering

Stanford University, CA 94305

bgirod,mkalman,yiliang,ruizhang@stanford.edu

Corresponding Author : Bernd Girod

Room 373, Packard Electrical Engineering

350 Serra Mall

Stanford, CA 94305

email: bgirod@stanford.edu

Telephone/Fax : (650) 724-6354 / (650) 725-8286

Abstract

Despite the well-known challenges of variations in throughput, delay, and packet loss over the Internet, video streaming has experienced phenomenal growth, owing to the extensive research in video coding and transmission. In this paper, we review several recent advances for channel-adaptive video streaming that, we believe, will benefit the design of video streaming systems in the future. Employed in different components of the system, these techniques have the common objective of providing efficient, robust, scalable and low-latency streaming video. First, by allowing the client to control the rate at which it consumes data, adaptive media playout can be used to reduce receiver buffering and therefore average latency, and provide limited rate scalability. Secondly, rate-distortion optimized packet scheduling, a transport technique, provides a flexible framework to determine the best packet to send given the channel behaviors, the packets' deadlines, their transmission histories, the distortion reduction associated with sending each packet, and the inter-packet dependencies. Thirdly, at the source encoder channel-adaptive packet dependency control can greatly improve the error-resilience of streaming video and reduce latency. Finally we address the specific additional challenges for wireless video streaming. We consider three architectures for wireless video and discuss the utility of the reviewed techniques for each architecture.

Keywords

Multimedia communication, video streaming, wireless video, error-resilient video coding, low latency, rate-distortion, scalability, source-channel coding.

I. INTRODUCTION

Since the introduction of the first commercial products in 1995, Internet video streaming has experienced phenomenal growth. Over a million hours of streaming media contents are being produced every month and served from hundreds of thousands of streaming media servers. Second only to the number-one Web browser, the leading streaming media player has more than 250 million registered users, with more than 200,000 new installations every day. This is happening despite the notorious difficulties of transmitting data packets with a deadline over the Internet, due to variability in throughput, delay and loss. It is not surprising that these challenges, in conjunction with the commercial promise of the technology, has attracted considerable research efforts, particularly directed towards efficient, robust, scalable and

low-latency video coding and transmission [1] [2].

A streaming video system has four major components: 1. The encoder application (often called the “producer” in commercial systems) that compresses video and audio signals and uploads them to the media server. 2. The media server that stores the compressed media streams and transmits them on demand, often serving hundreds of streams simultaneously. 3. The transport mechanism that delivers media packets from the server to the client for the best possible user experience, while sharing network resources fairly with other users. 4. The client application that decompresses and renders the video and audio packets and implements the interactive user controls. For the best end-to-end performance, these components have to be designed and optimized in concert.

The streaming video client typically employs error detection and concealment techniques to mitigate the effects of lost packets [3]. Unless forced by firewalls, streaming media systems do not rely on TCP for media transport but implement their own application level transport mechanisms to provide the best end-to-end delivery while adapting to the changing network conditions. Common issues include retransmission and buffering of packets [4], generating parity check packets [5], TCP-friendly rate control [6], and receiver-driven adaptation for multicasting [7]. New network architectures, such as DiffServ [8] and the path diversity transmission in packet networks [9], also fall into this category. The media server can help implementing intelligent transport mechanisms, by sending out the right packets at the right time, but the amount of computation that it can perform for each media stream is very limited due to the large number of streams to be served simultaneously. Most of the burden for efficient and robust transmission is therefore on the encoder application, which can not adapt to the varying channel conditions and must rely on the media server for this task. Rate scalable representations are very important to allow adaptation to varying network throughput without requiring computation at the media server.

Multiple redundant representations are an easy way to achieve this task, and they are widely used in commercial systems [4]. To dynamically assemble compressed bit-streams without drift problems, S-frames [10] and, recently, SP-frames [11] have been proposed. Embedded scalable video representations such as FGS [12] would be more elegant for rate adaptation, but they are still considerably less efficient, particularly at low bit-rates. Embedded scalable representations are a special case of multiple description coding of video that can be combined advantageously with packet path diversity [9] [13]. Finally, the source coder can trade-off some compression efficiency for higher error resilience [14]. For live encoding of streaming video, feedback information can be employed to adapt error resiliency, yielding the notion of channel-adaptive source coding. Such schemes have been shown to possess superior performance [15]. For pre-compressed video stored on a media server, these channel-adaptive source coding techniques can be effected through assembling sequences of appropriately precomputed packets on the fly.

In our opinion, the most interesting recent advances in video streaming technology are those that involve several system components jointly and react to the packet loss and delay, thus performing channel-adaptive streaming. In this paper, we review some recent advances in channel-adaptive streaming. As an example of a new receiver-based technique, we discuss adaptive media playout in Section II, which reduces the delay introduced by the client buffer and provides rate scalability in a small range. We then review rate-distortion optimized packet scheduling as the most important recent advance in transport mechanisms in Section III. An example of a channel-adaptive encoder/server technique we discuss is the new idea of packet dependency control to achieve very low latency as well as robustness in Section IV. All of these techniques are applicable for wireline as well as wireless networks. Architectures and the specific challenges arising for wireless video streaming are discussed in the concluding Section V.

II. ADAPTIVE MEDIA PLAYOUT

Adaptive Media Playout (AMP) is a new technique that can be used to reduce latencies in streaming systems that rely on buffering at the client to protect against random packet losses and delays. In these systems, the client buffers an amount of data before playout begins. While the likelihood of a buffer underflow, an event which causes media playout to halt, decreases as more data is stored, the latency experienced by the user increases. The client must therefore trade off buffer underflow probability and latency.

AMP, by giving the client some control over the rate at which its playout process consumes data, allows reduced buffer sizes and latencies for a given buffer underflow probability. With AMP, the client, without the involvement of the server, controls its data consumption rate by varying the speed at which it plays out media. For video, the client simply adjusts the duration that it shows each frame. For audio, the client performs signal processing to scale the signal in time while preserving its pitch [16]. Informal subjective tests have shown that slowing the playout rate of video and audio up to 25% is often un-noticeable, and that time-scale modification is subjectively preferable compared to halting playout or errors due to missing data [16] [17].

The buffering latency most noticeable to the user is pre-roll delay, the time that it takes for the buffer to fill with data and for playout to begin after the user makes a request. In today's commercial streaming products, these delays typically range from 5 to 15 seconds. An AMP-enabled client can reduce pre-roll delays by beginning playout with fewer frames of media stored in the buffer. Using slowed playout, the client can decrease the initial consumption rate so that the buffer occupancy increases over time. When a sufficient amount of media is buffered, playout can continue at normal speed.

Another noticeable latency is the constant buffering delay during live streaming or in two-way communication. By allowing both slowed and faster-than-normal playout,

an AMP-enabled client can reduce the mean delay experienced by the user, for a given probability of underflow. Slowed playout, used during bad channel periods to decrease the likelihood of an underflow, causes delay to increase over time. Therefore, during good channel periods the client must play media faster than normal to eliminate any delay that has accumulated. We have found that fast and slow playout allows a mean latency that is smaller than the constant delay that the user would experience in the case of fixed playout speed, for a given probability of underflow. The application was explored for the case of two-way voice communication in [16]. It can be easily extended to include video.

The Markov chain analysis and simulation results shown in Figs. 1 and 2 illustrate the latency reductions achievable with AMP. In both figures, the results are for a model of the packet network that randomly transitions between a good and a bad state. When the channel is in the good state, the mean number of frames that can be delivered to the client per frame period of the media sequence is given by λ_G . In the bad state, the number is given by λ_B . The mean durations of the good and bad states are T_G and T_B . We model the durations of the channel states and frame inter-arrival times as exponentially distributed random variables.

Fig. 1 illustrates pre-roll delay reduction with AMP. It is a plot of the required pre-roll time for a pre-stored media clip of a given length such that the media clip will play without interruption 99% of the time. These results are for an AMP policy which dictates that whenever the client buffer contains fewer than 10 seconds of media, playout frame periods will be stretched by a factor s . We see in the plot that by allowing playout to be slowed, less pre-roll time is needed. For instance, with slowdown factor $s = 1.25$, a 3.5 second pre-roll time will allow reliable streaming of programs longer than 1000 seconds, compared to the 7 second program allowable for the same pre-roll time without AMP.

Fig. 2 illustrates AMP's ability to reduce the mean buffering delay during live

streaming. The figure plots Mean Time Between Buffer Underflows (MTBBU) as a function of mean buffering delay. In this case the AMP policy dictates that frame periods are scaled by factor s when the number of frames in the buffer falls below a threshold, and by factor f when the number of frames in the buffer exceeds the threshold. In the plot we see, for example, that when $s = 1.25$ and $f = 0.75$, a mean latency of 5.25 seconds yields an MTBBU of 25 minutes compared to an MTBBU of 10 minutes for the same delay without adaptation ($s = f = 1.0$).

Another application of AMP employs it as a form of rate-scalability, allowing clients to access streams which are encoded at a higher source rate than their connections would ordinarily allow [18].

III. R-D OPTIMIZED PACKET SCHEDULING

The second advance that we review in this paper is a transport technique. Because playout buffers are finite, and because there are constraints on allowable instantaneous transmission rates, retransmission attempts for lost packets divert transmission opportunities from subsequent packets and reduce the amount of time that subsequent packets have to successfully cross the channel. A streaming media system must make decisions, therefore, that govern how it will allocate transmission resources among packets.

Recent work of Chou et al. provides a flexible framework to allow the rate-distortion optimized control of packet transmission [19] [20]. The system can allocate time and bandwidth resources among packets in a way that minimizes a Lagrangian cost function of expected rate and distortion. For example, consider a scenario in which uniformly sized frames of media are placed in individual packets, and one packet is transmitted per discrete transmission interval. A rate-distortion optimized streaming system decides which packet to transmit at each opportunity based on the packets' deadlines, their transmission histories, the channel statistics, feedback information, the packets' interdependencies, and the reduction in distortion yielded by each packet

if it is successfully received and decoded.

The framework assumes that opportunities to transmit packets occur at discrete intervals in time. At each transmission opportunity, the algorithm determines which packets to transmit by optimizing its transmission choices for the current opportunity jointly with a complete contingency plan for transmissions that it will make during a horizon of future opportunities. The contingency plan is comprised of a transmission policy π_l for each packet dictating whether packet l is transmitted or not at each transmission opportunity, contingent on feedback that is received. A transmission policy induces an error probability, $\epsilon(\pi_l)$, where an error is defined as the event that a packet does not arrive at the receiver by the time it is needed for playout. The policy also induces an expected number of times that the packet is transmitted, $\rho(\pi_l)$. The algorithm decides how it will allocate transmissions among the set of packets under consideration by finding the set of corresponding policies which minimize the cost function for those packets $J = D + \lambda R$, where $R = \sum_l \rho(\pi_l) B_l$ is the expected rate (B_l is the size in bytes of packet l) that will be incurred, and D is the expected reproduction distortion of the media stream, given the transmission choices contained in the set of policies. D , the expected distortion, is given by:

$$D = D_0 - \sum_l \Delta D_l \prod_{l' \preceq l} (1 - \epsilon(\pi_{l'})). \quad (1)$$

D_0 is the reproduction distortion of the group of media frames if no packets arrive, ΔD_l is the expected amount of distortion that is removed if packet l is decodable by its deadline, and the product term $\prod_{l' \preceq l} (1 - \epsilon(\pi_{l'}))$ is the probability that packet l is decodable. $l' \preceq l$ refers to the set of packets l' which must be present to decode packet l . The scheme assumes that loss probabilities for packets are independent, and that expected distortion reductions ΔD_l can be factored independently of the loss probabilities for individual packets.

Fig. 3 shows the rate-distortion improvement achievable for a two-layered, H.263+ SNR scalable mode encoding of the *Foreman* sequence, with a frame rate of 10 fps,

a GOP consisting of one I-frame followed by 9 P-frames, a mean source rate of 32 kbps for base layer alone, and a mean source rate of 64 kbps when combined with the enhancement layer. The results are for a channel model that assumes independent packet losses with loss probability 20%, and independent, Γ -distributed end-to-end delays with mean delay 50 ms. The traces in the figure plot PSNR as a function of rate for a nominal system and a system using R-D optimization. In the nominal system, the client requests retransmissions for packets that don't arrive by a time interval after they are expected, and the server transmits these requested packets with priority as long as the requested packet may still reach the client in time for playout. When the capacity of the channel falls below the source rate for the enhanced stream, the nominal system sends only the base layer packets. Both the nominal and optimized system use an initial pre-roll delay of 400 ms. In the figure, we see that by continuously optimizing its packet transmission choices, the optimized system makes use of the SNR and temporal scalability of the source encoding to finely tune the source rate to the available channel capacity, giving gains of up to 2.5 dB in quality, for this source encoding.

The framework put forth in [19] is flexible. Using the framework, optimized packet schedules can be computed at the sender [19], the receiver [20], or the proxy-server on the network edge [21]. The authors have also presented simplified methods to compute approximately optimized policies that require low computational complexity. Furthermore, the framework, as shown in [20], appears to be robust against simplifications to the algorithm and approximations to ΔD_l , the information characterizing the value of individual packets with respect to reconstruction distortion. Low complexity is important for server-based implementation, while robustness is important for receiver-based implementations, where the receiver makes decisions. We have recently extended Chou's framework to include adaptive media playout, such that each packet is optimally scheduled, along with a recommended individual playout dead-

line. For that, the distortion measure is extended by a term that penalizes time-scale modification and delay [22].

IV. CHANNEL-ADAPTIVE PACKET DEPENDENCY CONTROL

Even if advanced techniques like adaptive media playout and R-D optimized packet scheduling are used at the client side and at the transport layer, video streaming still typically exhibits much higher latencies than that of voice transmission over the Internet. This is due to the strong dependencies among video packets caused by interframe prediction. If a packet containing, say, one frame is lost, the decoding of all subsequent frames depending on the lost frame will be affected. Hence, in commercial systems, time is allowed for several retransmissions in order to guarantee the error-free reception of each frame, at the cost of higher latency. Advanced source coding schemes are thus desired to reduce latency while maintaining robustness.

Packet dependency control has been recognized as a powerful tool to increase error-robustness, and recently, to address the latency issue. Earlier work on this topic includes long-term memory (LTM) prediction for macroblocks for increased error-resilience [23], the reference picture selection (RPS) mode in H.263+ [24] and the emerging H.26L standard [25], and the video redundancy coding (VRC) technique [26]. These encoding schemes can be applied over multiple transmission channels for path diversity to increase the error-resilience [9], [27], [28].

In our recent work [29], to increase error-resilience and eliminate the need for retransmission, multiple representations of certain frames are pre-stored at the streaming server. A representation can be chosen in which only frames that will be received with high probability are used as reference frames. By doing so, we dynamically control the dependencies among packets to adapt to the varying channel conditions. With increased error-resilience, the need for retransmission can be eliminated. Because buffering is needed only to absorb the packet delay jitter, buffering delay can be reduced to a few hundred milliseconds.

Due to the trade-off between error-resilience and coding efficiency, we apply Optimal Picture Type Selection (OPTS) within an R-D framework, which considers video content, channel loss probability and channel feedback (e.g., ACK, NACK, or timeout). To code a frame, multiple attempts are made, including Intra-coding as well as Inter-coding using different reference frames in the long-term memory. For a particular attempt, v , the associated rate R_v and expected distortion \bar{D}_v are obtained to calculate the cost through a Lagrangian formulation

$$J_v = \bar{D}_v + \lambda R_v. \quad (2)$$

Here v indicates which of the previous frames is used for prediction, where $v = \infty$ denotes an I-frame. To find the expected distortion in (2), a binary tree model is employed which accounts for the channel loss rate and error propagation [29]. Note that the distortion includes both the quantization error and possible decoding mismatch error, which is calculated at the encoder side. For the Lagrange multiplier λ , we use $\lambda = 5e^{0.1Q}(\frac{5+Q}{34-Q})$, which is the same as λ_{mode} in H.26L TML 8 used to select the optimal prediction mode [25], [30], and Q is the quantization parameter used to trade off rate and distortion. The optimal picture type is selected such that the minimal R-D cost J_v is achieved

$$v_{opt} = \arg \min_{v=1,2,\dots,V,\infty} J_v, \quad (3)$$

where V is the length of the long-term memory.

This rate-distortion optimization applies to both the compression and streaming of live and pre-encoded video. However, a major challenge for streaming pre-encoded video is the potential mismatch due to the loss of the frame used for prediction. This mismatch error might propagate and lead to severe degradation in performance. Past work addressing this problem includes using S-frames [10], and SP-frames, as proposed for H.26L [11], [25]. However, both are achieved at the cost of a significant bit-rate increase.

We use a layered coding structure to avoid the mismatch errors. The coding structure consists of three layers as shown in Fig. 4: 1) *Layer I* contains I-frames that are spaced at intervals of T_{GOP} , the maximum length of a Group of Picture (GOP) which is a multiple of the long term memory length V . They serve as the lead of a GOP. 2) *Layer II* contains only two types of pictures: PV-frames (P frames using $v = V$) and I-frames. They are spaced at intervals of V , and serve as the lead of sub-GOP. Layer I and Layer II pictures are also referred to as *SYNC-frames*, which are only positioned at kV , where $k = 0, 1, 2, \dots$ (a video sequence starts from 0). The prediction dependency of SYNC-frames in a GOP is shown in Fig. 4. T_{GOP}/V versions of SYNC-frames in different phases are pre-stored, with the leading I-frame starting from different positions as shown in Fig. 4. The deployment of the SYNC-frames with multiple phases allows the server to insert an I-frame at *any* SYNC position (kV) to eliminate the mismatch error. The choice between continuing with a PV-frame in a GOP or switching to an I-frame is determined within the OPTS framework considering channel conditions and feedback. Once an I-frame is inserted, it starts a new GOP. 3) *Layer III* pictures are those within a sub-GOP following its lead. They can be either P-frames that use a previous frame in the same sub-GOP as reference, or occasionally I-frames depending on the video content. This limits the prediction dependency within a sub-GOP. Because there are T_{GOP}/V different phases of SYNC-frames, T_{GOP}/V different versions of Layer III pictures must be pre-stored.

Layer III frames are pre-encoded offline based on the video content and the expected channel condition, while SYNC-frames are dynamically assembled during delivery to respond to the channel feedback. The sub-GOP lead determines which set of Layer III pictures to send. Pre-stored multiple versions of the bitstreams enable mismatch-free assembly, though achieved at the cost of extra disk storage. The layered coding also provides temporal scalability for rate control and facilitates interactive functions such as random access, fast-forward or fast-reverse.

Simulation results in [29] compare the performance of 1) the proposed channel-adaptive OPTS scheme, and 2) a simple scheme that uses normal P-frames with periodic I-frames to combat packet loss (referred to as the *P-I* scheme), where T_{GOP}/V phases of the sequence are also pre-stored, and feedback-induced I-frame insertion at certain positions is allowed. Fig. 5 shows the R-D performance of sending 230 frames of *Foreman* sequence, respectively, at 30fps over the channels with 10% loss rate. Modified H.26L TML 8.5 [25] is implemented. Distortion at different channel loss rates is shown in Fig. 6 for *Foreman* encoded at approximately the same 200 Kbps using different schemes. The gain in PSNR ranges from 0.5 dB to 1.6 dB, depending on the channel loss rate.

It is observed that although no retransmission is used, good quality is still maintained for typical video sequences sent over lossy channels. Thus the high robustness achievable through packet-dependency control can be used to eliminate the need for retransmission, leading to latencies similar to those for Internet voice transmission.

V. CHALLENGES OF WIRELESS VIDEO STREAMING

In our previous discussion, we have not differentiated between video streaming for the wireline and the wireless Internet. Increasingly, the Internet is accessed from wireless, often mobile terminals, either through wireless LAN, such as IEEE 802.11, or 2.5G or 3G cellular networks. It is expected that in 2004, the number of mobile Internet terminals will exceed the number of fixed terminals for the first time. Wireless video streaming suffers from the same fundamental challenges due to congestion and the resulting best-effort service. Packets still experience variable delay, loss, and throughput. Channel-adaptive techniques as discussed above are important to mitigate these problems.

The mobile radio channel, however, introduces specific additional constraints, and many of the resulting challenges still hold interesting research problems. Fading and shadowing in the mobile radio channel lead to additional packet losses, and hence

TCP-style flow control often results in very poor channel utilization. Frame sizes of wireless data services are usually much smaller than the large IP packets preferable for video streaming, hence fragmentation is necessary. Since the loss of any one fragment knocks out an entire IP packet, this effectively amplifies the loss rate of the wireless link. An obvious remedy is to use ARQ for the radio link, trading off throughput and delay for reliability. Most, but not all mobile data services operate in this way.

Another objection to using IP for streaming over mobile radio links concerns the RTP/UDP/IP encapsulation overhead that can use up a significant portion of the throughput of the expensive wireless link. Moreover, mobility management in IP is lacking, and mobile IP protocols that employ further encapsulation might be even more wasteful. Header compression [31], however, can very efficiently overcome this problem and will be widely deployed in future radio systems.

Three alternative architectures for wireless video streaming are shown in Fig. 7. The end-to-end architecture in Fig. 7 (a) preserves the Internet paradigm of stateless routing with connection-oriented services implemented in the terminals. Channel-adaptive streaming methods, as discussed above, would be implemented in the client and the server only. We need to distinguish systems with ARQ on the radio link and lossy systems. In order to solve the problem of sharing bandwidth fairly both in the wireline and the lossy wireless links, reliable loss differentiation algorithms (LDA) are required that can distinguish loss due to congestion and a deteriorating wireless channel. Some promising research is underway, but the proposed techniques are still limited [32]. ARQ in the radio link can avoid wireless losses altogether, but reduces throughputs and increases delay. For streaming applications where delay is not critical, radio link ARQ is superior.

Fig. 7 (b) shows an architecture with a proxy server separating the wireless and wireline portion of the network. Instead of connecting to the back-end streaming media server directly, the client connects to the proxy server, which in turn connects

to the streaming media server. The proxy is responsible for pre-fetching and buffering packets, such that they are available when the mobile client needs them. Channel-adaptive streaming techniques can now be applied to each of the two connections separately. The proxy server might also implement simple transcoding to reduce the bit-rate or increase error resilience for low-delay applications.

Fig. 7 (c) shows an architecture where a gateway between the wireline and wireless part of the network marks the territory of the Internet. For the wireless link, an integrated wireless media protocol, tailored to the needs of wireless audio and video transmission, is used. This integrated wireless media protocol could even be a circuit-switched multimedia protocol stack, such as H.324M [33]. Channel-adaptive streaming techniques would be used between the gateway and the streaming media server, while packet-oriented streaming media techniques, such as dynamic packet scheduling, might not be applicable to the wireless link. With H.324M, error-resilience of the video stream is important, as is rate scalability or rate control to accommodate variable effective throughput even on a nominally fixed-rate link. The 3G-PP consortium has evolved the ITU-T recommendation H.324M into 3G-324M, which also supports MPEG-4 video, in addition to H.263v2, for conversational services.

The streaming architecture in Fig. 7 (c) is actually being implemented by some companies, but it appears to be a short-term solution. The segregation of the world into wireline and wireless terminals is far too serious a drawback. Establishing and tearing down a circuit for each video stream is cumbersome and wasteful, particularly considering that a packet-switched always-on connection will soon be widely available in 2.5G and 3G systems. The open architecture of IP-solutions as shown in Fig. 7 (a) and (b) will undoubtedly prevail.

REFERENCES

- [1] M. R. Civanlar, A. Luthra, S. Wenger, and W. Zhu (eds.), *Special Issue on Streaming Video, IEEE Trans. on Circuits and Systems for Video Technology* 2001; 11(3).
- [2] C. W. Chen, P. Cosman, N. Kingsbury, J. Liang, and J. W. Modestino (eds.), *Special Issue on Error*

- Resilient Image and Video Transmission, IEEE Journal on Selected Area in Communications* 2001; 18(6).
- [3] Y. Wang, and Q. Zhu, "Error control and concealment for video communication: a review," *Proceedings of the IEEE* 1998; 86(5):974-997.
 - [4] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video coding for streaming media delivery on the Internet," *IEEE Trans. on Circuits and Systems for Video Technology* 2001; 11(3):269-281.
 - [5] W. Tan, and A. Zakhor, "Video multicast using layered FEC and scalable compression," *IEEE Trans. on Circuits and Systems for Video Technology* 2001; 11(3):373-387.
 - [6] W. Tan, and A. Zakhor, "Real-time Internet video using error resilient scalable compression and TCP-friendly transport protocol," *IEEE Trans. on Multimedia* 1999; 1(2):172-186.
 - [7] S. McCanne, M. Vetterli, and V. Jacobson, "Low-complexity video coding for receiver-driven layered multicast," *IEEE Journal on Selected Areas in Communications* 1997; 15(6):983-1001.
 - [8] J. Shin, J. Kim, and C.-C. J. Kuo, "Quality-of-service mapping mechanism for packet video in differentiated services network," *IEEE Transactions on Multimedia* 2001; 3(2):219-231.
 - [9] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee, "On multiple description streaming with content delivery networks," *IEEE Infocom*, July 2002.
 - [10] N. Färber, and B. Girod, "Robust H.263 compatible video transmission for mobile access to video servers," *Proc. IEEE International Conference on Image Processing, ICIP'97*; 2:73-76; Santa Barbara, CA, USA.
 - [11] M. Karczewicz, and R. Kurceren, "A proposal for SP-frames," Proposal to H.26L, Jan. 2001.
 - [12] M. van der Schaar, and H. Radha, "A hybrid temporal-SNR fine-granular scalability for Internet video," *IEEE Trans. on Circuits and Systems for Video Technology* 2001; 11(3):318-31.
 - [13] Y. Wang, M. Orchard, V. Vaishampayan, and A. R. Reibman, "Multiple description coding using pairwise correlating transforms," to appear in *IEEE Trans. on Image Processing*.
 - [14] R. Zhang, S. L. Regunathan, and K. Rose, "Video coding with optimal inter/intra-mode switching for packet loss resilience," *IEEE Journal on Selected Areas in Communications* 2000; 18(6):966-976.
 - [15] B. Girod, and N. Färber, "Wireless video," in A. Reibman, M.-T. Sun (eds.), *Compressed Video over Networks*; Marcel Dekker, 2000.
 - [16] Y. J. Liang, N. Färber, and B. Girod, "Adaptive playout scheduling using time-scale modification in packet voice communication," *Proc. IEEE International Conference on Acoustic Speech and Signal Processing, ICASSP'01*, Salt Lake City, May 2001.
 - [17] E. G. Steinbach, N. Färber, and B. Girod, "Adaptive playout for low Latency video streaming," *IEEE Proc. International Conference on Image Processing (ICIP'01)*, Thessaloniki, Greece, Oct. 2001.
 - [18] M. Kalman, E. Steinbach, and B. Girod, "Adaptive playout for real-time media streaming," *Proc. IEEE International Symposium on Circuits and Systems, ISCAS'02*, Scottsdale, AZ, May 2002. Invited Paper.
 - [19] P. A. Chou and Z Miao, "Rate-distortion optimized streaming of packetized media," *IEEE Transactions on Multimedia*, February 2001. Submitted. <http://research.microsoft.com/~pachou>.
 - [20] P. A. Chou and A. Sehgal, "Rate-distortion optimized receiver-driven streaming over best-effort networks," *Proc. Packet Video Workshop*, Pittsburg, PA, April 2002.
 - [21] J. Chakareski, P. A. Chou, and B. Girod, "Rate-distortion optimized streaming from the edge of the

- network,” *IEEE Workshop on Multimedia Signal Processing*, St. Thomas, US Virgin Islands, December 2002. Submitted.
- [22] M. Kalman, E. Steinbach, and B. Girod, “R-D optimized media streaming enhanced with adaptive media playout,” in *Proc. International Conference on Multimedia and Exhibition, Lausanne, Switzerland*, Aug. 2002.
- [23] T. Wiegand, N. Färber, and B. Girod, “Error-resilient video transmission using long-term memory motion-compensated prediction,” *IEEE Journal on Selected Areas in Communications* 2000; 18(6):1050-1062.
- [24] ITU-T Recommendation H.263 Version 2 (H.263+), *Video coding for low bitrate communication*, Jan. 1998.
- [25] ITU-T Video Coding Expert Group, *H.26L Test Model Long Term Number 8*, July 2001, online available at <ftp://standard.pictel.com/video-site/h26L/tml8.doc>.
- [26] S. Wenger, G. D. Knorr, J. Ott, and F. Kossentini, “Error resilience support in h.263+,” *IEEE Transactions on Circuits and Systems for Video Technology* 1998; 8(7):867-877.
- [27] S. Lin, S. Mao, Y. Wang, and S. Panwar, “A reference picture selection scheme for video transmission over ad-hoc networks using multiple paths,” *Proc. IEEE International Conference on Multimedia and Expo (ICME)*, Aug. 2001.
- [28] Y. J. Liang, E. Setton and B. Girod, “Channel-adaptive video streaming using packet path diversity and rate-distortion optimized reference picture selection,” submitted to *IEEE Fifth Workshop on Multimedia Signal Processing*, Dec. 2002.
- [29] Y. J. Liang and B. Girod, “Rate-distortion optimized low-latency video streaming using channel-adaptive bitstream assembly,” *accepted by IEEE International Conference on Multimedia and Expo*, Aug. 2002.
- [30] T. Wiegand and B. Girod, “Lagrange multiplier selection in hybrid video coder control,” in *Proc. IEEE International Conference on Image Processing, ICIP’01*; 3:542-545; Thessaloniki, Greece.
- [31] C. Bormann et al., “Robust Header Compression (ROHC),” *IETF RFC 3095*, <http://www.ietf.org/rfc/rfc3095.txt>.
- [32] S. Cen, P. C. Cosman, and G. M. Voelker, “End-to-end differentiation of congestion and wireless losses,” *SPIE Multimedia Computing and Networking (MMCN2002)*, San Jose, CA, Jan. 2002.
- [33] N. Färber, B. Girod, and J. Villasenor, “Extensions of the ITU-T Recommendation H.324 for error resilient video transmission,” *IEEE Communications Magazine* 1998; 36(6):120-128.

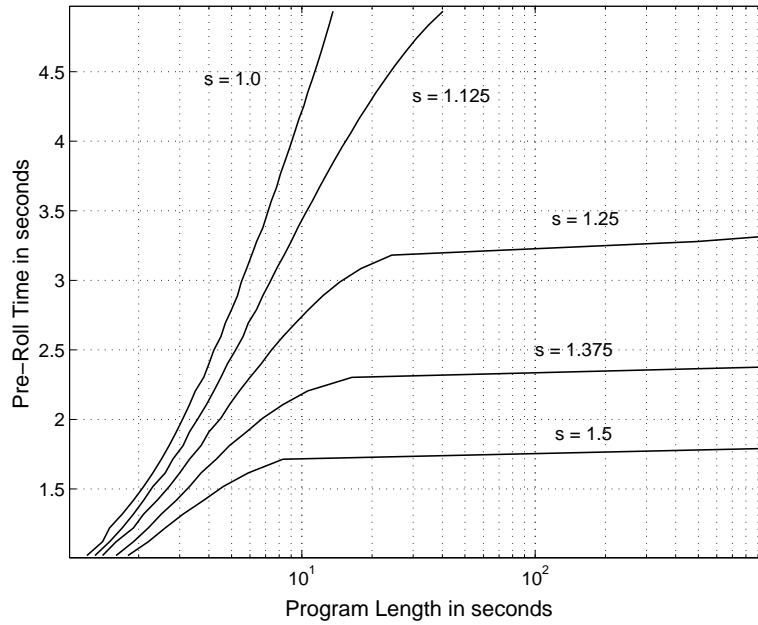


Fig. 1. Required pre-roll time as a function of program length for 99% reliability, if frame periods are stretched by factor s whenever buffer level falls below 10 seconds of media. Channel model parameters are $\lambda_G = 1.09$, $\lambda_B = 0.4$, $T_G = 20sec$, and $T_B = 2sec$.

Bernd Girod, et al., “Advances in Channel-adaptive Video Streaming”

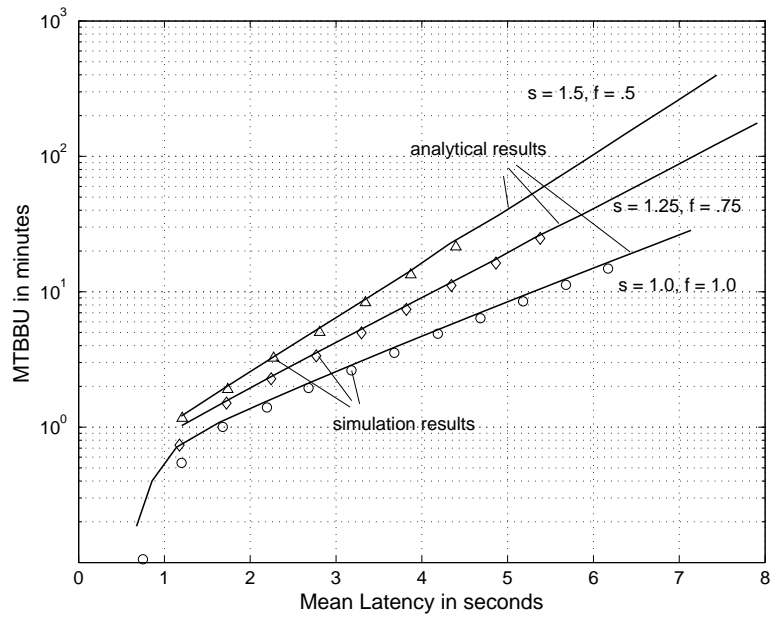


Fig. 2. Mean time between buffer underflows as a function of mean latency when frame periods are scaled by factor s when the buffer level is below a threshold and scaled by factor f when the level exceeds the threshold. Channel model parameters are $\lambda_G = 1.33$, $\lambda_B = 0$, $T_G = 28.5\text{sec}$, and $T_B = 1.5\text{sec}$.

Bernd Girod, et al., “Advances in Channel-adaptive Video Streaming”

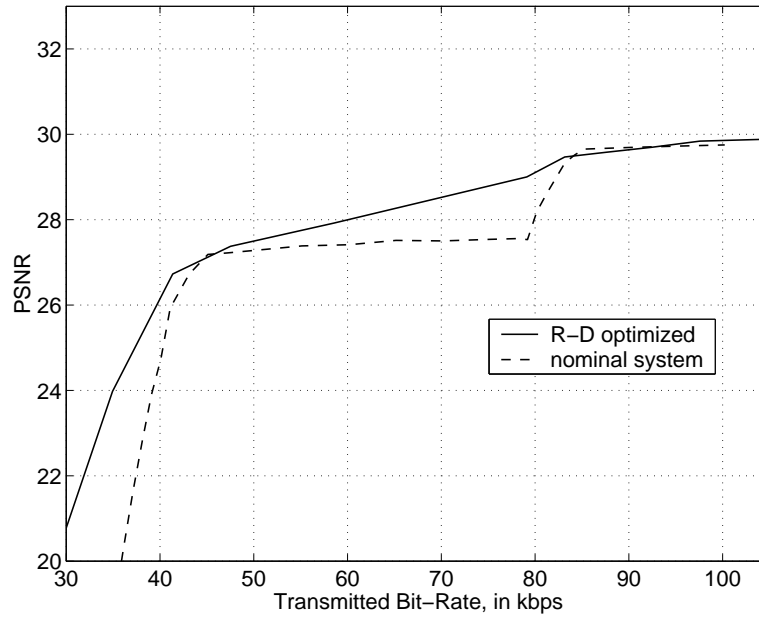


Fig. 3. PSNR vs. transmitted bit-rate for a nominal streaming system that uses ARQ and for a streaming system that uses R-D optimized transmission scheduling. The results are for an H.263+, SNR scalable encoding of *Foreman*.

Bernd Girod, et al., “Advances in Channel-adaptive Video Streaming”

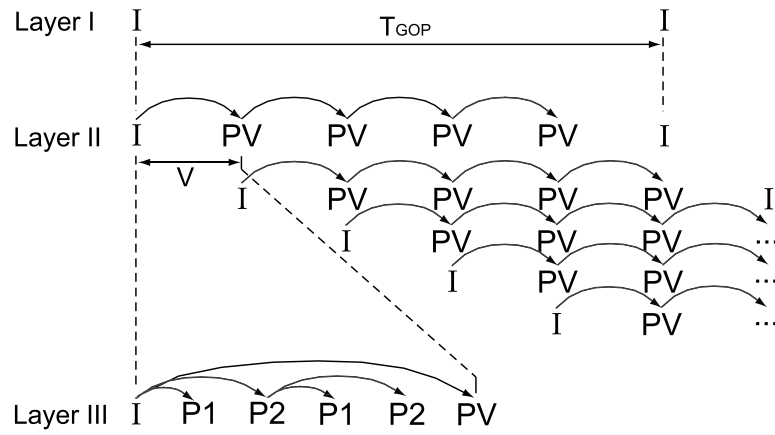


Fig. 4. Layered coding structure with coding restrictions. $T_{GOP} = 25$, $V = 5$.

Bernd Girod, et al., "Advances in Channel-adaptive Video Streaming"

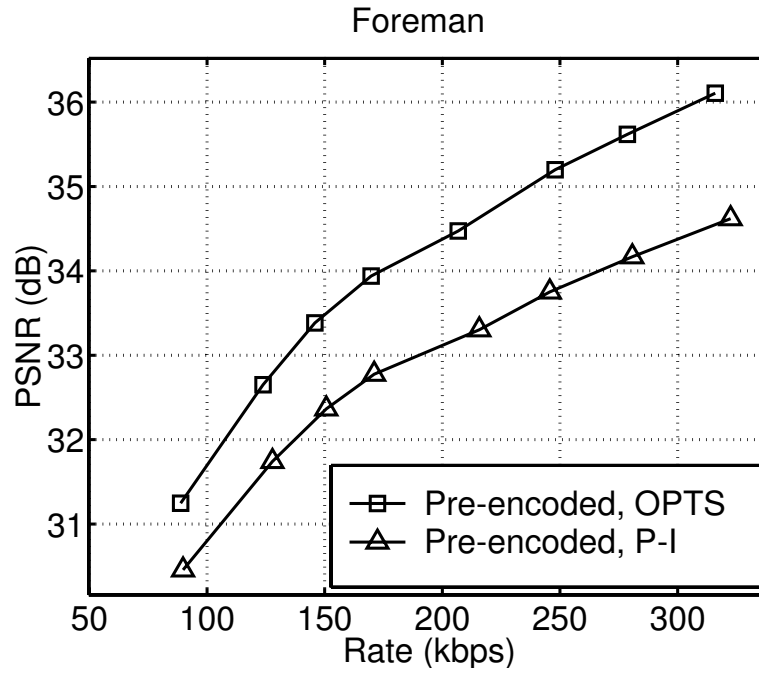


Fig. 5. RD performance for *Foreman* sequence, 10% packet loss, 5 frames feedback delay, $V = 5$.

Bernd Girod, et al., “Advances in Channel-adaptive Video Streaming”

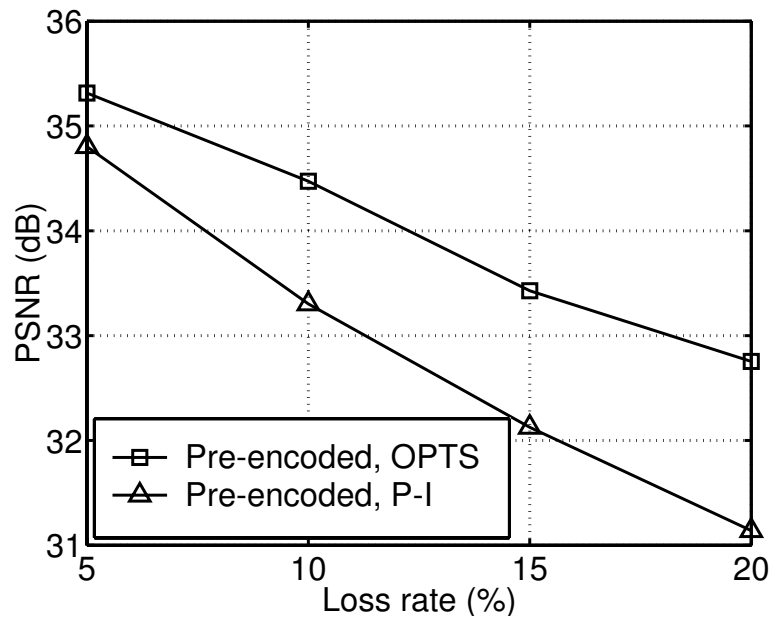


Fig. 6. Distortion at different channel loss rates. *Foreman* sequence. The bitrate is 200 Kbps.

Bernd Girod, et al., "Advances in Channel-adaptive Video Streaming"

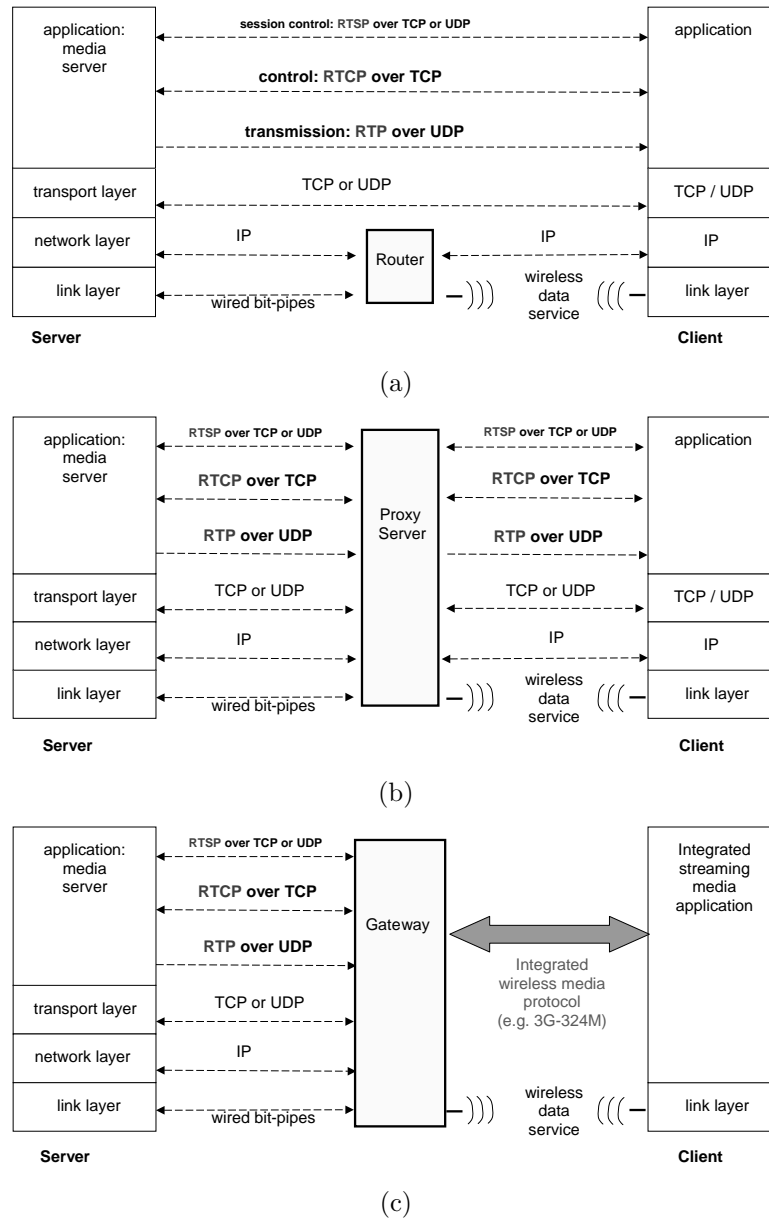


Fig. 7. Wireless streaming architectures: (a) end-to-end, (b) proxy server, (c) gateway with integrated media protocol.

Bernd Girod, et al., “Advances in Channel-adaptive Video Streaming”