H.264/AVC Over IP

Stephan Wenger

Abstract—**H.264 is the ITU-T's new, nonbackward compatible video compression Recommendation that significantly outperforms all previous video compression standards. It consists of a video coding layer (VCL) which performs all the classic signal processing tasks and generates bit strings containing coded macroblocks, and a network adaptation layer (NAL) which adapts those bit strings in a network friendly way. The paper describes the use of H.264 coded video over best-effort IP networks, using RTP as the real-time transport protocol. After the description of the environment, the error-resilience tools of H.264 and the draft specification of the RTP payload format are introduced. Next the performance of several possible VCL- and NAL-based error-resilience tools of H.264 are verified in simulations.**

Index Terms—**Data partitioning, flexible macroblock ordering (FMO), H264, RTP, slice interleaving.**

I. Introduction

H.264 is the denomination of ITU-T's most recent video codec Recommendation, which is also known as ISO/IEC14496-10 or, less formally, as MPEG-4 Advanced Video Codec (AVC). It is a product of the joint video team (JVT) consisting of the members of MPEG and the ITU's Video Coding Experts Group. H.264 consists of a video coding layer (VCL) and a network adaptation layer (NAL). The VCL consists of the core compression engine, and comprises syntactical levels commonly known as the block-, macroblock-, and slice level. It is designed to be as network independent as possible. Its main design goals, implementation, and performance are reported elsewhere in this special issue [1]. The VCL contains several coding tools that enhance the error resilience of the compressed video stream. Those tools are briefly introduced later in this paper.

The NAL adapts the bit strings generated by the VCL to various network and multiplex environments. It covers all syntactical levels above the slice level. In particular, it includes mechanisms for:

- the representation of the data that is required to decode individual slices (data that resides in picture and sequence headers in previous video compression standards);
- the start code emulation prevention;
- the support of supplementary enhancement information (SEI);
- the framing of the bit strings that represent coded slices for the use over bit-oriented networks.

As a result of this effort, it has been shown that the NAL design specified in the Recommendation is appropriate for the adaptation of H.264 over RTP/UDP/IP, H.324/M, MPEG-2 transport, and H.320. An integration into the MPEG-4 system framework is also well on its way to standardization.

The main motivation for introducing the NAL, and its separation from the VCL is twofold. First, the Recommendation defines an interface between the signal processing technology of the VCL, and the transport-oriented mechanisms of the NAL. This allows for a clean design of a VCL implementation—probably on a different processor platform than the NAL. Second, both the VCL and the NAL are designed in such a way that in heterogeneous transport environments, no source-based transcoding is necessary. In other words, gateways never need to reconstruct and re-encode a VCL bit stream because of different network environments. This holds true, of course, only if the VCL of the encoder has provisioned the stream for the to-be-expected or measured, end-to-end transport characteristics. It is, for example, the VLC responsibility to segment the bit stream into slices appropriate for the networks in use, to use sufficient nonpredictively coded information to cope with erasures, and so forth.

The paper is organized in six main sections. Section II reviews the general constraints that apply to the transmission of compressed video over IP. More specifically, the target applications with their specific constraints and the protocol environment of the IP-based network are discussed. Readers familiar with the characteristics of IP networks and RTP packetization schemes for video may want to skip this section. Section III discusses the error-resilience tools available in H.264. Many of these tools, such as slice structuring and intra macroblock placement, are well known from earlier standards and previous research. However, some new technology has also emerged that deserves additional discussion. The tools of Section III are equally suited to IP and wireless networks, and are discussed in some depth, whereas the error concealment and the encoder mechanisms for error resilience, especially the loss-aware rate-distortion (R-D) optimization, are introduced in [2]. Section IV goes over the concepts of the RTP packetization for H.264, as it stands in discussions in the IETF as of April 2003. It is believed that the final RTP payload specification will be implementing most or all of the tools discussed in this section—however, it is likely that some of the details will be changed. In Section V, the error-resilience tools in the VCL and NAL and the RTP payload specification are exposed to source video material and a network simulation in order to verify their combined performance. Emphasis is placed on conversational, video-conferencing-like applications that disallow the use of most channel-based error protection schemes due to delay constraints, as outlined in Section II.

II. Video Transmission Over IP

This section discusses the environment to which an IP-based H.264 codec may be exposed. After going over the technical

Manuscript received December 12, 2001; revised May 9, 2003.

The author is with Teles AG, Berlin, Germany, and also with the Technical University of Berlin, 10587 Berlin, Germany (e-mail: stewe@cs.tu-berlin.de).

Digital Object Identifier 10.1109/TCSVT.2003.814966

characteristics of key applications for IP-based video, the currently used protocol infrastructure and their characteristics are introduced.

A. Applications

Before discussing the transmission of video over IP, it is necessary to take a closer look at its intended applications. The nature of which determines the constraints and the protocol environment with which the video source coding has to cope.

Using IP as a transport, three major applications can currently be identified.

- *Conversational applications, such as videotelephony and videoconferencing.* Such applications are characterized by very strict delay constraints—significantly less than one second end-to-end latency, with less than 100 ms as the (so far unreachable) goal. They are also limited to point-to-point or small multipoint transmissions. Finally, they imply the use of real-time video encoders and decoders, which allow the tuning of the coding parameters in real-time, including the adaptive use of error-resilience tools appropriate to the actual network conditions, and often the use of feedback-based source coding tools. However, the use of real-time encoders also limits the maximum computational complexity, especially in the encoder. Low delay constraints further prevent the use of some coding tools that are optimized for high-latency applications, such as bipredicted slices.
- *The download of complete, pre-coded video streams.* Here, the bit string is transmitted as a whole, using reliable protocols such as ftp [3] or http [4]. The video coder can optimize the bit stream for the highest possible coding efficiency, and does not have to obey restrictions in terms of delay and error resilience. Furthermore, the video coding process is normally not a real-time process; hence, computational complexity of the encoder is also a less critical subject. Most of the traditional video coding research somewhat implies this type of application.
- *IP-based streaming.* This is a technology that, with respect to its delay characteristics, is somewhere in the middle between download and conversational applications. There is no generally accepted definition for the term “streaming”. Most people associate it with a transmission service that allows the start of video playback before the whole video bit stream has been transmitted, with an initial delay of only a few seconds, and in a near real-time fashion. The video stream is either pre-recorded and transmitted on demand, or a life session is compressed in real-time—often in more than one representation with different bit rates—and sent over one ore more multicast channels to a multitude of users. Due to the relaxed delay constraints when compared to conversational services, some high-delay video coding tools, such as bipredicted slices, can be used. However, under normal conditions, streaming services use unreliable transmission protocols, making error control in the source and/or the channel coding a necessity. The encoder has only limited—if any—knowledge of the network conditions and has to adapt the error resilience tools to a level that most users would find acceptable. Streaming video is sent from a single server, but may be distributed in a point-to-point, multipoint, or even broadcast fashion. The group size determines the possibility of the use of feedback-based transport and coding tools.

This paper is mostly concerned with conversational services, because here techniques from both the source coding and the channel coding must be employed, and their interaction can be shown. In addition, most research within JVT with respect to IP-transport was performed assuming such an application. Many of the discussions also apply to a streaming environment. Readers primarily interested in download-type applications should refer to papers that are concerned with coding efficiency in this special issue [5].

IP networks can currently be found in two flavors: unmanaged IP networks, with the Internet as its most prominent example, and managed IP networks such as the wide-area networks of some long-distance telephony companies. An emerging third category could also be addressed: wireless IP networks based on the third-generation mobile networks. (Please see [2] in this Special Issue for an in-depth discussion.)

All three network types have somewhat different characteristics in terms of the maximum transfer unit size (MTU size), the probability for bit errors in packets, and the need to obey the the Transmission Control Protocol (TCP) traffic paradigm.

1) MTU Size: The MTU size is the largest size of a packet that can be transmitted without being split/recombined on the transport and network layer. It is generally advisable to keep coded slice sizes as close to, but never bigger than, the MTU size, because this: 1) optimizes the payload/header overhead relationship and 2) minimizes the loss probability of a (fragmented) coded slice due to the loss of a single fragment on the network/transport layer and the resulting discarding of all other fragments belonging to the coded slice in question (by the network/transport layer protocols). The end-to-end MTU size of a transmission path between two IP nodes is very difficult to identify, and may change dynamically during a connection. However, most research assumes MTU sizes of around 1500 bytes for wireline IP links (because of the maximum size of an Ethernet packet). In a wireless environment, the MTU size is typically considerably smaller—most research including JVT’s wireless common conditions assume an MTU size of around 100 bytes.

2) Bit Errors: Bit-error probabilities of today’s wireline networks are so low that, within the scope of this work, they can be safely ignored. (Please see [2] for a discussion on how the H.264 test model handles the significantly higher bit error rates found in wireless networks.)

3) Rate Control and TCP Traffic Paradigm: Since the big Internet Meltdown of the late 1980s, the transport protocol TCP [6], which is used to carry most Internet content such as email and Web traffic, obeys the so-called TCP traffic paradigm [7]. It would be beyond the scope of this paper to discuss it in detail but, in short, the TCP traffic paradigm mandates that a sender reduces its sending bit rate to half (as a result of an adjustment of the TCP buffer size) as soon as it observes a packet loss rate above a certain threshold. Once the packet loss rate drops below the threshold (plus some hysteresis), the sender may slowly increase its sending bit rate again, until the packet loss rate is once again too high and the whole process restarts. This simple, yet effective regulation mechanism prevents the overload of routers and ensures reasonable fairness among all senders. It should be emphasized that due to this architecture, packet losses and network dictated bit rates for connections are a very basic feature of all best-effort IP networks, and not a result of a network failure of some type. This mechanism was introduced as a congestion control method and works well if the packet losses are the result of congestion. In wireless networks, where packet losses due to shortcomings of the link layer protocols are more common, this algorithm does not yield the optimal results, but it is nevertheless one of the foundations of IP networks in general and the Internet in particular, and should be observed by all network elements to prevent future congestion related meltdowns.

Until now, most media transport protocol implementations simply ignore the TCP traffic paradigm. Media rate control schemes that take TCP-friendliness into account have been reported in academia [8], but have not yet seen wide deployment in the field—mostly because, so far, no one on the public Internet is "enforcing" the TCP traffic paradigm. This has not yet resulted in an observable problem for the operation of the Internet as a whole, only because the percentage of such traffic is comparatively low—and the conforming TCP applications behave "extra fairly" and reduce their sending bit rate by a higher than fair amount, in order to free the bit rate for the unfair real-time senders. With the increased popularity of real-time services, and especially, real time media transmission, it can be anticipated that Internet Service Providers (ISPs) will also start to enforce a TCP-like congestion control for media traffic as well, in order to assure a good quality of service for the regular TCP traffic. Private managed networks often do not have this problem, as they are either over-provisioned or use bandwidth allocation techniques. The same is true for the (also privately operated) wireless network and their wireline infrastructure parts. Hence, an IP-based video sending device needs to support both congestion control aware and unaware transmission schemes to ensure a high quality user experience.

A final remark on the relationship of congestion control and error-resilient coding: the goal of the congestion control mechanisms is to reduce the network load when higher error rates are observed. Error-resilient video coding tends to add redundancy to the bit stream to cope with the higher error rates—and, thus, increase the bit rate and the network load if the same quality level should be kept. As a result, the two main tools of the community that are used to combat errors use contradicting approaches. Until now, the only solution for this problem that has gained wide acceptance has been the reduction of the quality of video in the source coder (or switching to a lower bit rate simulcast stream)—by reducing the sending frame rate, picture size, picture quality and, in the worst case, dropping the video transmission as a whole.

B. Protocol Environment

Luckily, for conversational and streaming applications there is only one single commonly used protocol hierarchy.

1) Physical and Link Layer: IP networks may operate over a variety of physical and link layer protocols and are generally designed to abstract from those underlying protocols. There is no need for a discussion of these two layers.

2) Network Layer: IP: On the network layer, IP networks obviously use the Internet Protocol IP [9]. Many pages could be filled with the design considerations, the actual design, and the properties of IP, and literally hundreds of papers and books have been published on the topic. For the purpose of this work, it is sufficient to say that IP packets are transported individually from the sender through a set of routers to the receiver. Splitting and re-combing of service data units (SDUs) larger than the MTU size is handled by IP as well. The transmission time of a packet varies from packet to packet, and routers are free to discard packets at any time—this condition is observed by the receiver as a packet loss. Hence, IP offers a so-called best effort service only. The IP header is 20 bytes in size and protected by a checksum to ensure its integrity. No protection of the payload is performed. The maximum size of an IP packet allowed by the protocol specification is 64 kbytes, but this size is rarely used because of MTU size constraints—please see Section II-A-I above.

3) Transport Layer: User Datagram Protocol (UDP): On the transport layer, IP networks commonly employ two protocols: the TCP [6] and the UDP [10]. Both include features common to transport protocols, such as application addressing through the port number, and error control for the payload.

TCP offers a byte-oriented, guaranteed transport service, which is based on re-transmission and timeout mechanisms for error control. Due to its unpredictable delay characteristics it is not suitable for real-time communication.

UDP, on the other hand, offers a simple, unreliable datagram transport service. The UDP header contains a checksum, which can be used to detect and remove packets containing bit errors—the mode of operation generally used. Apart from this, UDP offers the same *best effort* service as IP does: packets may be lost, duplicated, or re-ordered on their way from the source to the destination. With UDP, if further error control schemes are deemed necessary, these need to be provided at a higher layer [also referred to as *application layer framing (ALF)* [11]]. The UDP header is 8 bytes in size.

4) Application Layer Transport: RTP: Such an application layer framing is implemented in the real-time transport protocol (RTP) [12]. RTP is typically used above IP/UDP. It is session oriented, and a session is associated with a transport address—the combination of the IP address and the UDP port number. Each RTP packet consists of an RTP header, optional payload headers, and the payload itself. The RTP header contains the following.

- The sequence number, which is incremented by one for each packet sent in a session and used for packet-loss detection.
- The timestamp that contains timing information relative to the establishment of the session. Timestamps are normally used to determine the precise moment for media reproduction, but also for purposes such as the synchronization of media streams carried in more than one session. For video,

the timestamp is usually generated using the sampling instant (capture time).

- The payload type, which identifies the media codec of the payload. For a few audio codecs, a direct mapping of the Payload Type integer to a media codec exists. For example, G.711 audio in a-law format is payload type 8 [13]. However, for most modern media codecs including H.264, the association between the payload type and the media coding must be established dynamically, per session, using a control protocol mechanism.
- The marker bit, which is normally set for the very last packet of a group of packets that have the same time stamp, e.g., belonging to the same video picture. It can be employed to quickly detect the end of a group of related packets without having to wait for the next packet to arrive.
- Some administrative information that is used mostly in conjunction with intelligent network entities such as media mixers and translators which are outside the scope of this paper.

The basic RTP header has a size of 12 bytes. The combined size of the IP/UDP/RTP header is $20+8+12=40$ bytes. This number is generally used later when discussing payload/overhead constraints. However, it should be noted that, at the edge of the IP network, header compression techniques can be used which allow reducing the header size considerably. Within the core network, the header size is unavoidable unless tunneling techniques are used, and these have their own problems [14].

An RTP sender operates normally using a very simple algorithm. The bit stream to be transported is divided into packets of reasonable size, ideally at locations that allow independent reconstruction. For many more powerful media-coding schemes including all video codecs, an additional payload header precedes this bit stream, whose format is specified in an RTP payload specification—please see Section IV. For the RTP header, the timestamp is set to the capture time, the sequence number is increased by one, the payload type and the marker bit are set according to their respective descriptions as above, and the packet is sent through UDP.

At the receiver, the sequence number of the RTP header allows the detection of packet losses in order to bring the packets into their original order, and to remove duplicated packets. A so-called “jitter buffer” is maintained, which is used to compensate for the different transmission times of UDP packets. Packets that arrive too late to be accepted into the jitter buffer are removed and observed as a packet loss by the media decoder. Therefore, the size of this jitter buffer must be tuned carefully according to the current network conditions and the application’s needs—if it is too big, it adds unnecessary delay, if it is too small, the loss rate, as observed by the media decoder, is increased—even at a constant network packet loss rate.

The RTP specification contains an accompanying simple control protocol. This protocol can be employed to inform an encoder about the network and transmission path characteristics as observed by the decoder, so that an encoder optimized for such use could adapt the error resilience and transmission bit rate in real-time.

5) Media-Unaware RTP Payload Specifications: There are currently a few techniques, often described as media unaware RTP payload specifications, or meta-payloads, which can be employed to reduce the loss rates as observed by the media decoder. They work by sending redundant information, which stands in fundamental contrast to the concepts behind the TCP traffic paradigm. Hence, such schemes should be used only to implement application-layer uneven error protection by protecting only a few of the packets. Moreover, all these techniques incur some additional delay, which makes them difficult to use, or even inapplicable, to low-delay applications. Three schemes should be introduced briefly, as they are relevant for the transmission of compressed video.

- Packet duplication is a most basic technique where the sender sends a to-be-protected RTP packet more than once and, thus, raises the probability for the packet to arrive at the receiver. Since packet duplications may also happen due to the internal operations of the network, all RTP receivers are prepared to remove duplicated packets—there is neither a need to announce the capability nor any changes in the decoding system. In the simulations of Section V, packet duplication is used to protect the most important bits of a video stream, with very positive results.
- Packet-based forward error correction (FEC), as specified in RFC 2733 [15], operates by calculating an XOR checksum over a number of to be protected packets, and sending the resulting bits as redundant information. In practice, packet-based FEC is not used in conversational applications due to the additionally incurred delay, but it is believed to be a valuable tool for streaming. (See [2] for a more in-depth discussion.)
- Audio redundancy coding (ARC, RFC2198 [16]) is, despite its name, a technique that is helpful to protect any data stream, including video. Each packet consists of the headers, the regular packet’s payload, and a redundant representation of the previous packet’s payload. H.264 can make use of this technique in conjunction with data partitioning—see below.

6) Application-Layer Control Protocols: Control protocols, such as H.245 [17], SIP [18] with SDP [19], or RTSP [20], are used to announce the availability of a media stream, to establish the (virtual or physical) connections, to negotiate the capabilities of the sender/receiver(s), and to control a running session. In most papers on video coding, they are ignored as not being relevant for the media transmission itself.

In H.264, however, bit streams are not necessarily self contained because all the information that affects more than one coded slice can be sent either out-of-band, e.g., though the control protocol, or in-band for applications that do not have a control channel, such as an MPEG-2 transport stream [21]. (See Section III-E the discussion on parameter sets for more information.) This paper cannot discuss, in depth, the mapping of out-of-band parameter set transmissions to the various control protocols, but readers may refer to [22] for some initial ideas on how an SDP-like syntax (which could be utilized in SIP, SAP, and RTSP environments) for such a transmission would look.

So far, the standardization community has not yet finalized any mapping scheme that could be referred to.

III. Error-Resilience Tools

As all video codecs developed over time, H.264 also includes a number of error-resilience tools. As in previous video coding standards, their application and adaptation is chosen by the encoder. As indicated below, many of those tools were present in previous video compression schemes as well, and therefore, do not need to be introduced here in much detail. However, a few tools are either completely new in standard-based video compression or are implemented in an innovated way, and hence require more attention.

In the older video compression standards (H.261, H.263, MPEG-1 Part 2 and MPEG-2 Part 2), the following error-resilience tools were already available:

- different forms of picture segmentation (slices, GOBs);
- placement of Intra MBs, intra slices, and intra pictures.

The more recent video compression standards, including the later versions of H.263 as well as some profiles of MPEG-4 Part 2, implemented a few more tools:

- reference picture delection (with and without feedback, picture, GOB/dlice, or MB-based);
- data partitioning.

H.264 introduces three new error-resilience tools, namely:

- parameter sets;
- flexible macroblock ordering (FMO);
- redundant slices (RS).

All tools mentioned above are discussed in Sections III-A–G; however, especially for the older tools, the author expects some familiarity of the reader with the technology. (See, for example, [23] and [24].)

A. Intra Placement

Intra placement on the macroblock, slice, or picture level, is used primarily to combat drifting effects. There are not many new constraints with respect to these tools in H.264. The following aspects, however, deserve attention.

- H.264 allows Intra macroblock prediction even from predictively coded (Inter) macroblocks. This feature is helpful for coding efficiency, but is counterproductive to the re-synchronization property of Intra coding. The ConstrainedIntraPrediction Flag on the sequence level, when set, prevents this form of prediction and restores the re-synchronization property of Intra information.
- H.264 has two forms of slices that contain only Intra macroblocks: Intra slices and IDR slices. IDR slices must always form a complete IDR picture—that is, all slices of an IDR picture must be IDR slices, and an IDR slice can only be part of an IDR picture. An IDR picture invalidates all short-term reference memory buffers, and, hence, has a stronger re-synchronization property than a picture that contains only Intra slices (which would, at the first glance, correspond in its re-synchronization property to an intra picture of older video compression standards). An Intra picture cancels drift for the duration of that picture, but if subsequent pictures reference information of pictures older than the Intra picture then drift would be re-established even in case of an error-free transmission of the Intra and all subsequent pictures.

The test model for error-prone environments uses a loss-aware rate/distortion optimized coder that is discussed in detail in [2], and sets the ConstrainedIntraPrediction flag. Earlier experiments based on H.263 [25] suggest that simpler algorithms, such as random or pseudo-random intra placement also yield acceptable results, at a relatively small bit rate penalty [26]. However, even applications using such a simpler algorithm will typically have to set the ConstrainedIntraPrediction flag.

B. Picture Segmentation

H.264 supports picture segmentation in the form of slices. A slice is formed by an integer number of MBs of one picture. No constraints are imposed on the encoder regarding the number of macroblocks in a slice—a slice can encompass as little as a single macroblock or as much as all macroblocks of a picture. However, every macroblock in a picture must be coded in exactly one slice (see the discussions of RSs later for a single exception). Macroblocks are assigned to slices in raster scan order, unless FMO is used (see Section III-F).

As in older video coding standards, slices interrupt the in-picture prediction mechanisms. Since all information traditionally found in the picture header is considered to be available at the decoder—which is ensured by the parameter set mechanism discussed later—the availability of one coded slice implies that the MBs of that slice can be reconstructed.

The main motivation for slices is the adaptation of the coded slice size to different MTU sizes, but they can also be used to implement schemes such as interleaved packetization (see [27] for details).

C. Reference Picture Selection

Reference picture selection, regardless of whether it is on a per picture, per slice, or per macroblock basis, can be used as an error-resilience tool in the same way it is used in H.263 [28]. In feedback-based systems, the encoder receives information about lost or damaged picture areas, and can react by choosing older—known as correct reference MBs for prediction—instead of using more expensive intra information. For systems without feedback, video redundancy coding may be employed [29]. An analysis of the efficiency of feedback-based error resilience when using H.264 can be found in [2].

D. Data Partitioning

Normally, all symbols of a macroblock are coded together in a single bit string that forms a slice. Data partitioning, however, creates more than one bit strings (called partitions) per slice, and allocates all symbols of a slice into an individual partition that have a close semantic relationship with each other.

In H.264, three different partition types are used.

- *Header information, including MB types, quantization parameters, and motion vectors.* This information is the most important, because without it, symbols of the other partitions cannot be used. This partition type is called type A partition in H.264.

- *The Intra Partition, called type B partition.* It carries Intra CBPs and Intra coefficients. The type B partition requires the availability of the type A partition of a given slice to be useful. In contrast to the Inter information partition (below), the Intra information can stop further drift and, hence, is more important than the Inter Partition.
- *The Inter Partition, called type C partition.* It contains only Inter CBPs and Inter coefficients but is, in many cases, the biggest partition of a coded slice. Inter Partitions are the least important because their information does not re-synchronize the encoder and the decoder. In order to be used, they require the availability of the type A partition, but not the type B partition.

When data partitioning is used, the source coder puts symbols of different types to three different bit buffers. Furthermore, the slice size must be adjusted in such a way that the biggest partition does not lead to a packet bigger than the MTU size. For this reason, it is the source coder and not the NAL that has to implement data partitioning.

In the decoder, all partitions need to be available to start standard-conformant reconstruction. However, if the Inter or the Intra partitions are missing, the available header information can still be used to improve the efficiency of error concealment. More specifically, due to the availability of the MB types and the motion vectors, a comparatively high reproduction quality can be achieved, as it is only the texture information that is missing.

E. Parameter Sets

Calling parameter sets an error-resilience tool is not entirely appropriate—they are generally used in all H.264 bit streams and hence described elsewhere in this special issue [1]. In short, the sequence parameter set contains all information related to a sequence of pictures (defined as all pictures between two IDR pictures), and a picture parameter set contains all information related to all the slices belonging to a single picture. Multiple different sequence and picture parameter sets can be available at the decoder in numbered storage positions. The encoder chooses the appropriate picture parameter set to use by referencing the storage location in the slice header of each coded slice. The picture parameter set itself contains a reference to the sequence parameter set to be used.

The intelligent use of the parameter set mechanism greatly enhances error resilience. The key of using parameter sets in an error-prone environment is to ensure that they arrive reliably, and in a timely fashion at the decoder. They can, for example, be sent out-of-band, using a reliable control protocol, and early enough, so that the control protocol time to get them to the decoder before the first slices that references that new parameter set arrives over the real-time communication channel [22].

Alternatively, they can be sent in-band, but with appropriate application layer protection (e.g., by sending multiple copies, so to enhance the probability that at least one copy arrives at the destination). A third option is that an application hard-codes a few parameter sets in both encoder and decoder, which would be the only operation points of the codec.

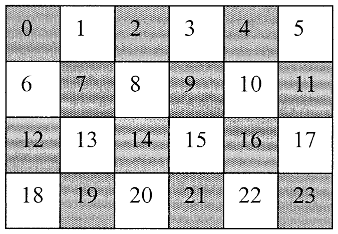

Fig. 1. A picture with a size of 6×4 MBs and two slice groups. The shaded MBs belong to slice group 0, the white MBs to slice group 1. Obviously, when losing one of the two slice groups, each lost (inner) MB has four neighboring MBs which can be used to conceal the lost information.

F. Flexible Macroblock Ordering

Flexible macroblock ordering, available in the Baseline and Extended, but not in the Main profile, allows to assign MBs to slices in an order other than the scan order (please see [30]–[32] for an in-depth discussion of FMO). To do so, each MB is statically assigned to a slice group using a macroblock allocation map (MBAmap). Within a slice group, MBs are coded using the normal scan order. In-picture prediction mechanisms, such as Intra prediction or motion vector prediction, however, are only allowed if the spatially neighboring MBs belong to the same slice group. To illustrate this relationship, please consider Fig. 1. All MBs of the picture are allocated either slice group 0 or slice group 1, depicted in grey and white, respectively, in a checker-board fashion. Assume that the picture is small enough to fit into two slices, one encompassing all macroblocks of slice group 0, the other encompassing all MBs of slice group 1. Assume further that, during transmission, the packet containing the information of slice group 1 got lost. Since every lost MB has several spatial neighbors that belong to the other slice, an error-concealment mechanism has a lot of information it can employ for efficient concealment. Experiments have shown that, in video conferencing applications with CIF-sized pictures, and at loss rates up to 10%, the visual impact of the losses can be kept so low that it takes a trained eye to identify the lossy environment—an efficiency level that was not achievable before with source-coding based tools. The price of the use of FMO is a somewhat lower coding efficiency (because of the broken in-picture prediction mechanisms between non-neighboring MBs) and, in highly optimized environments, a somewhat higher delay.

G. RSs

RSs allow an encoder to place, in addition to the coded MBs of the slice itself, one or more redundant representations of the same MBs into the same bit stream. The key difference between a transport based redundancy, such as packet duplication as discussed in Section II-B-V), and the use of RSs is that the redundant representation can be coded using different coding parameters. For example, the primary representation could be coded with a numerically low QP (and hence in good quality), whereas the RS could be coded with a numerically high QP (hence, in a much coarser quality, but also utilizing fewer bits). A decoder reacts to RSs by reconstructing only the primary slice, if it is available, and discarding the RS. However, if the primary slice is missing (e.g., as the result of a packet loss), the RS can be reconstructed. RSs were introduced to support highly error-prone mobile environments, but are equally efficient in IP-based environments (see [33]).

IV. RTP Packetization

This section discusses the draft RTP payload specification for H.264. At the time of writing, this document contains several sections that will change, for example advanced use forms of aggregation packets and fragmentation. Hence, only the most basic features are discussed here—these are the features which are later used in the simulations section of this paper. However, before discussing the packetization scheme itself, the NAL unit concept of H.264 shall be recapitulated.

A. H.264's NAL Unit Concept

Earlier video compression standards were always centered around the concept of a bit stream. Higher layer syntax elements were separated by start codes to allow re-synchronization to the bit stream in case of corruption—be it the result of an erasure or of bit errors. H.264, when employing its optional Annex B, also allows such a framing scheme, primarily to support a few legacy protocol environments such as H.320 or MPEG-2 transport. The RTP packetization, however, employs the native NAL interface that is based on NAL units (NALUs).

A NALU is a byte string of variable length that contains syntax elements of a certain class. There are, for example, NALUs carrying a coded slice, a type A, B, C data partition, or a sequence or picture parameter set. Each NALU consists of a 1-byte header with three fixed-length bitfields, and a variable number of bytes containing the coded symbols. The header has the following format:

The NALU type (T) is a 5-bit field that characterizes the NALU as one of 32 different types. Types 1–12 are currently defined by H.264. Types 24 to 31 are made available for uses outside of H.264, and the RTP payload specification employs some of these values to signal aggregation and fragmentation packets—see below. All other values are reserved for future use by H.264. The nal_reference_idc (R) can be employed to signal the importance of a NAL unit for the reconstruction process. A value of 0 indicates that the NAL unit is not used for prediction, and hence could be discarded by the decoder or by network elements without risking drifting effects—even though with a negative impact for the user (the same impact that the discarding of B-frames has in previous video compression standards). Values higher than 0 indicate that the NALU is required for a drift-free reconstruction, and the higher the value, the higher the impact of a loss of that NALU (in the encoder's opinion) would be. Network elements can use the nal_reference_idc to protect important packets more forcefully than less important ones. The forbidden_bit, finally, is specified to be zero in H.264 encoding. Network elements can set this bit to 1 when they identify bit errors in the NALU. The forbidden_bit is primarily useful in heterogeneous network environments (such as combined wireline/wireless environments), where some decoders may be prepared to operate on NALUs containing bit errors and others do not. Consider a wireless to wireline gateway, with a non-IP protocol environment on the wireless side and a bit-error-free IP environment on the wireline side. Furthermore, consider a decoder to be connected to the wireline side of that gateway. Assume a NALU arrives on the wireless side that fails a checksum test. The gateway could choose to remove that NALU from the NALU stream, so to make sure that all NALUs arriving at the receiver would be bit-error free. It could also forward the known-as-corrupt NALU in an UDP packet to the receiver—however, that packet would pass the UDP checksum test (as it was packetized by the gateway for the transport over the reliable wireline link), and a decoder would not have a chance to prepare itself for the corrupted data. In this case, the gateway sets the forbidden_bit. An intelligent decoder could now attempt to reconstruct the NALU (knowing that it may include bit errors) whereas a less intelligent decoder would simply discard such a NALU.

B. Packetization Design Constraints

The design constraints for the H.264 RTP payload specification design can be summarized as follows.

1) It should have low overhead, so that MTU sizes of 100 bytes (or less) to 64 kbytes are feasible.
2) It should be easy to distinguish "important" from "less important" RTP packets, without decoding the bit stream carried in the packet.
3) The payload specification should allow the detection of data that became undecodable due to other losses, without a need to decode the bit stream. Gateways, for example, should be able to detect the loss of a type A partition and if desired, react to this by not sending the type B and type C partitions.
4) It should support NALU fragmentation into multiple RTP packets.
5) It should support NALU aggregation—more than one NALU to be transported in a single RTP packet.

Since the NAL concept of H.264 was designed with IP networks in mind, many of the design constraints are already reflected in the H.264 design. In particular, there is no need for an additional payload header when using simple packetization forms—the NALU header co-serves as the RTP header. This feature satisfies the design constraints 1, 2, and 3. To support constraints 4 and 5, code points of the NALU type are employed that were reserved by JVT for external use. These code points are employed to mimic a special form of NALU type, and thus to minimize the overhead even for those advanced features. Both the fragmentation and the aggregation features are defined in such a way that media aware network elements can perform fragmentation, aggregation, and the respective inverse functions without parsing the media stream beyond the NALU header byte.

C. Simple Packetization

All forms of packetizations of H.264 NALUs are called "simple packetization" schemes that put exactly one NALU in one RTP packet. The packetization rules for this mode are indeed very simple: put a NALU (including its NALU header, which co-serves as the payload header) into the payload of an RTP packet, set the RTP header values as defined in the RTP specification itself [12], and send it. Ideally, the VCL should never generate NALUs that are bigger than the MTU size, in order to avoid IP-layer fragmentation. This can easily be achieved by employing slices. However, as long as the NALUs are smaller than 64 kbytes, IP performs the fragmentation and the recombination of fragmented packets—hence even by using simple packetization most pre-recorded NALU streams can be conveyed.

At the receiver's side, duplicated packets, as identified by the RTP sequencing information, are discarded, and the NALUs are extracted from the RTP packets. The timing information of the RTP packets is also required for a time-exact rendering, and overwrites the timing information that can be part of an SEI NALU. The baseline and extended profiles allow the out-of-order decoding of slices. Thus, re-ordering of packets in the jitter buffer is not required. When the main profile is used (which implies that out-of-order slices are not allowed) the re-ordering of packets, by using the RTP sequencing information, is required. The introduction of a decoding order number (DON) concept is currently under discussion in the IETF. This would formalize the reaction of the receiver to the out-of-order reception of NALUs that belong to different pictures.

When uneven protection schemes are available in the network, using data partitioning is a very efficient way to improve the error resilience without adding much overhead [34]. The data partitions are protected according to their importance for reconstruction—partition A is protected better than partition B, and partition C is sent best-effort. The partitions B and C are useless when the corresponding type A partition is missing. Hence, an RTP receiver could discard RTP packets containing lonely type B and/or C partitions. This feature is particularly useful for media-aware gateways. When a gateway senses a type A partition is being lost, it is free to discard type B and C partitions belonging to the same slice and, hence, lowers the network burden and eases the congestion of the network.

D. NALU Fragmentation

There may be cases, e.g., when using pre-encoded content, where the encoder cannot react to the MTU size demands of the underlying networks and many NALUs bigger than the MTU size would have to be transmitted. IP layer fragmentation is available to handle this case for NALU sizes up to 64 kbytes. However, when relying on IP fragmentation, obviously no application layer protection of the fragments can be used. This would render most of the concepts related to uneven protection schemes, e.g., those based on data partitioning or to the use of nonreferenced NALUs, inefficient or even counter-productive.

For this reason, and for the sheer fact that (without IP-V6 Jumbograms) UDP datagrams cannot be bigger than 64 kbytes—a coded slice size that may simply be too small for applications such as digital cinema—an application layer fragmentation scheme is part of the RTP packetization scheme.

The fragmentation scheme is currently undergoing fine-tuning in the IETF. The following basic features are expected to be part of the ratified RFC.

- The fragments of a fragmented NALU are transmitted in ascending order of RTP sequence numbers—no RTP packets containing (parts of) other NALUs are allowed between the fragments of a given NALU.
- A signaling mechanism exists which indicates the first and the last fragment of a fragmented NALU.
- Another signaling mechanism is available that allows the detection of a sequence of lost fragments between NALU boundaries.

0	1	2	3	4	5	6	7
T					R		F

Fig. 2. NALU header.

E. NALU Aggregation

Some H.264 NALUs, e.g., SEI NALUs, or the parameter set NALUs are typically very small—a few bytes at most. It would be helpful to aggregate them with other NALUs into a single RTP packet so to reduce the overhead for the IP/UDP/RTP headers. For this reason, an aggregation scheme has been introduced.

Two basic types of aggregation packets exist:

- single-time aggregation packets (STAPs) contain NALUs with identical timestamp;
- multi-time aggregation packets (MTAPs) can contain NALUs with different timestamps.

STAPs are normally used in low-delay environments, where any form of interleaving of information belonging to multiple pictures is undesirable. They are of a relatively simple design—a 1-byte header of the same format as the NALU header (see Fig. 2), indicating that this packet is an STAP, is followed by one or more STAP aggregation units. Each aggregation unit consists of a length indication of the following NALU, and is followed by the NALU itself. The use of STAPs is assumed in some of the simulations in Section V.

MTAPs are more useful in high delay environments such as streaming. They allow for some sophisticated packetization schemes that greatly enhance the performance of H.264 based streaming—see, for example, [33]. The basic structure of an MTAP is similar to an STAP; however, the aggregation unit also contains a timestamp (coded relative to the RTP timestamp in order to save bits), and a decoding order number that is used to indicate the decoding order of the NALUs in the MTAP. Since the mechanisms behind both concepts are still undergoing design changes, no further discussion seems to be appropriate at this time.

V. Current Performance Figures

This section contains the results of some simulations performed with the current release of the TML software, JM 1.0 [35] and JM 1.7 in case of the FMO simulations [36], and a set of network simulation tools that are generally used by JVT. Note that JM 1.0 and JM 1.7 are roughly equivalent in the R-D performance, and therefore, it is believed the comparisons made later are valid.

A. JVT's Testing Environment

In order to efficiently and objectively compare the performance of newly proposed coding tools to the reference design, a set of common testing conditions (CCs) is maintained by JVT, which have to be employed by all performance tests. Each CC document is concerned with a certain application and its transport environment. The CC document relevant to this work is [37]. When testing video over IP, the packet lossy nature of the IP network has to be simulated. For this, the CC document mandates a very simple tool, which discards some IP packets depending on the information available in a supplied error pattern file. Four error pattern files with average packet loss rates of roughly 3%, 5%, 10%, and 20% are used—see [38] for details on their characteristics and how they were generated.

The current CC document acknowledges the fact that a feedback-based operation can largely increase the reproduced picture quality. However, due to the difficulties in the simulation environment, only simplex video transmission is assumed. This restriction makes the CCs applicable to all video over IP scenarios including those without an available feedback channel (i.e., broadcast). Please note the assumption that some form of quality monitoring of the forward video channel is available. On IP networks, using RTP this is the case through RTCP, which scales to very large receiver groups of tens of thousands of receivers. In those environments where true simplex communication is necessary, a reasonable error rate has to be "guessed" by the service provider, and taken into account when adapting the error resilience (in the video coding or on the transport channel, or both combined).

B. Simulations

Due to space and time constraints, only a small subset of the simulation results required by the CCs are presented in Section V-B-I. More results can be found in the JVT document archives [39].

1) Simulation Environment: Simulations are presented using two sequences.

- Foreman, in QCIF size, and with a frame skip of three pictures, corresponding to a target frame rate of 7.5 fps. The bit rate for the complete packet stream, including the complete overhead for IP/UDP/RTP is kept close to, but below 64 kbit/s. This is achieved by choosing the lowest numerical quantizer value that fits this requirement. For Foreman, a total of 100 coded frames corresponding to 13.33 s of video are used.
- Paris, in CIF size, and with a frame skip of 1 picture, corresponding to a target frame rate of 15 fps. The gross bit rate is kept below 384 kbit/s. Here, a total of 200 coded frames, corresponding to 10 s of video, are tested.

The codec release JVT1.0, dated 2/5/02, is employed. To generate the RTP stream using the simple packetization (one NALU per RTP packet), no changes to the software were necessary. However, small stand-alone tools for tasks such as the duplication of Type-A partitions or to implement aggregation packets for interleaved packetization were required, and implemented as a set of C programs.

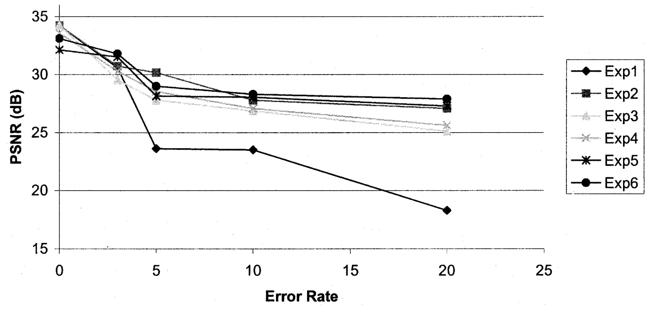

Fig. 3. Simulation results for the foreman sequence.

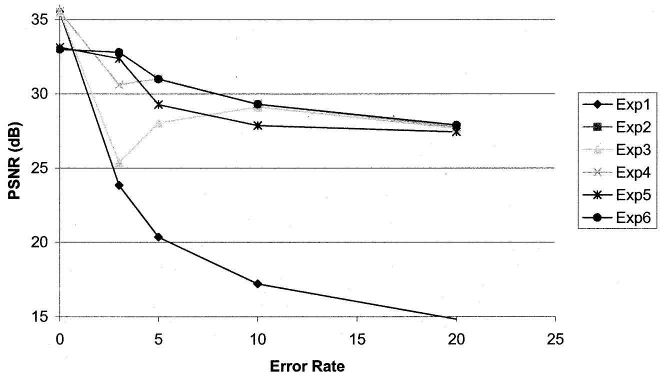

Fig. 4. Simulation results for the paris sequence.

The simulation parameters were set as follows, except where noted otherwise in the description of the experiments:

- no Intra GOB refresh used;
- QP kept constant at a value to match the bit rate requirements;
- motion resolution 1/4 pel;
- all macroblock types enabled;
- only one single reference frame—no multiframe prediction used;
- no B-slices used;
- UVLC-type entropy coding (note that later releases of the H.264 draft use the term CA-VLC for an optimized version of the UVLC entropy coding);
- error concealment as discussed in [2] is used.

With these common parameters, the encoder settings were chosen so that they are at the lower end of the complexity scale of the encoder, and quite realistic for a real-time implementation (see [40] for an evaluation of the real-time properties of such a simple setting).

2) Experiments Performed: The wealth of error-resilience tools of both the VCL and the NAL, and their dependence on each other, make it impossible to evaluate each tool individually. Reasonable combinations have to be chosen in order to get some impression of the performance of H.264 over IP.

A total of six experiments were performed.

Fig. 3 plots the results of all six experiments for the Foreman sequence, and Fig. 4 for the Paris sequence, both at the packet loss rates of 0%, 3%, 5%, 10%, and 20%.

For the diagrams, a regular PSNR measurement was used. The PSNR was only calculated for the pictures that were actually reconstructed (in whole or in part with error concealment).

Against the recommendation in the common conditions document, no penalty was introduced for lost pictures. If such were the case, one could expect that the quality for the Experiments 1, 2, and 4 would have been significantly worse than indicated in the diagrams, because in all those experiments a relatively large number of complete pictures would have gotten lost, giving no hint to the decoder that they existed. In Experiment 5, the header partition was protected well enough that here few, if any, pictures would have been lost.

1) Experiment 1: One picture, one packet, without any error-resilience tools applied

In Experiment 1, the encoder was configured to generate optimum coding efficiency for an error-free environment. Only a single slice per picture was used, and no bits were spent on robustness. Consequently, the results are the best in the error free and the worst in all error-prone test runs. It is fair to say that at packet loss rates above 3% unprotected H.264 video becomes unusable. This is in line with research on other video coding schemes that employ inter picture prediction.

2) Experiment 2: One picture, one packet, with intra macroblock refresh

This experiment used a loss-aware R-D optimization process, as discussed in detail in [2] for optimal intra placement of macroblocks, and a one picture, one packet approach. Experiment 2 yields very good objective results for the Foreman sequence. However, this is accomplished by using a very high amount of intra MBs of significant size, which leads to a coarser quantizer value and, hence, less spatial detail. The IP/UDP/RTP header overhead is kept to the theoretical minimum for low-delay applications. Experiment 2 was not performed for Paris, because most packets would exceed the MTU size, leading to a significantly higher observed packet loss rate. Clearly, at bit rates of 384 kbits/s and only 15 fps target frame rate (which yields an average coded picture size of 3.2 kbyte) coded video needs some form of picture fragmentation, e.g., slices.

3) Experiment 3: Slices

In Experiment 3, it was tried to adjust the IP/UDP/RTP header overhead to a similar amount as used by the Experiments 4, 5, and 6, by employing at least three slices per picture. This has the welcome side effect that the picture loss rate is minimized as it is very likely that at least one slice of every coded picture is received. For Foreman, three packets per picture were used, for Paris the number was variable because slices were filled until the MTU size was reached, but typically 2–4 slices per picture were used. Due to the additional header overhead fewer bits were available to code video, and, hence, a coarse quantizer had to be selected to stay within the bit rate requirements. For the Foreman sequence, this led to unfavorable PSNR results. However, subjectively, and in contrast to the objective PSNR results, for the Foreman sequence Experiment 3 is superior to Experiment 2 due to the constant high frame rate. For the Paris sequence, Experiment 3 leads to the second best PSNR results (next to FMO), and the subjective quality is comparable to the one of the two other experiments with applied error-resilience tools.

4) Experiment 4: Slice interleaving

Experiment 4 uses slice Interleaving, a technique that has turned out to be successful for other video coding standards, and is used commercially in several video telephony systems [22]. At least two packets per picture are used, and for the Paris sequence, many pictures have taken four packets to stay within the MTU size constraints. Each slice encompasses a single line of macroblocks, similar to the way slices are commonly used in MPEG-2. Even numbered macroblock lines, and odd numbered macroblock lines are transported in different packets. This allows for efficient concealment when only one of the packets is lost, since the macroblocks above and below a lost macroblock are available for concealment. Subjectively and for the Paris sequence, slice Interleaving produced very good results, which is also reflected in the PSNR values. This is particularly true because no lost frames could be observed in this case. For Foreman, the penalty due to the increased header overhead, when compared to Experiment 2, led to slightly inferior results.

5) Experiment 5: Data partitioning

Experiment 5 must be distinguished from the other experiments because it does not employ any intra refresh for error-resilience purposes. It relies completely on the superior error concealment mechanisms that are available when it can be made sure (or almost sure) that at least the header partition arrives. This is achieved by sending the RTP packet with the header partition twice (for the 3% error rate case) or 3 times (for the 5%, 10%, and 20% error rate cases). Of course, the quantizer is adapted so that the complete packet stream, including the multiple header partitions, fits into the bit rate budget. No loss-aware R-D optimization is used for this Experiment. Hence, the computational encoding complexity is significantly lower than those in the other experiments. With the JM1.1 reference software, the encoding speed for Experiment 5 is more than twice higher than the other experiments.

For the Foreman sequence, superior objective and subjective results can be reported. For Paris, they are at least competitive. Undoubtedly, the performance of Experiment 5 could be further increased when using an optimized intra refresh method for this scheme, but thus far time constraints have disallowed the implementation of such a mechanism.

6) Experiment 6: Flexible macroblock ordering

As a completely new coding tool, a good performance is expected, and indeed FMO outperforms all other tested mechanisms at high error rates (where it is designed for). This, of course, to a large part is the result of the error concealment mechanisms in JM1.7, which allows the use of a very low intra rate. In Experiment 6, the R-D optimized intra placement is not used, instead a simple, pseudo-random intra placement is employed, with an intra rate that was one third of the packet loss rate (e.g., 3.3% for 10% packet loss rate). This intra rate was chosen as a result of a large set of experiments on test sequences distinct from those used in the common conditions. This results in the use of far less intra than in many other experiments, yielding more available bits for the inter coefficients and thus in a numerically smaller QP and more visible detail. Since two slice groups are employed, the minimum number of slices per picture is two as well. For Paris, however, most pictures took four packets and the associated overhead, which resulted in a very low picture loss rate. The quality of the reconstructed video could be described as "crisper", however some artifacts not frequently seen in error-resilient video coding could be observed. Unfortunately, such artifacts are only visible in motion sequences, hence it was refrained from reproducing reconstructed pictures in this paper.

VI. Conclusions

H.264, in addition to all its coding-efficiency oriented features, introduces a set of new tools that improve the error resilience. In particular, the concepts of parameter sets and NALUs on the NAL, and FMO and data partitioning on the video coding layer have been shown to significantly improve the performance of H.264 in the challenging best-effort IP environment. Well-known tools, such as enhanced reference picture selection and Intra placement can also be employed to enhance the reproduced quality in an error-prone environment and, in some cases, they come close to the performance level of the new tools. However, all the intelligence of an encoder employing the old and new coding tools would not be at all helpful, without a good mapping of the video bits to the payload of RTP packets. The draft RTP payload specification for H.264, which closely integrated with the H.264 NAL, provides guidance for the packetization. It also adds some transport-level, low overhead mechanisms that allow for efficient fragmentation and aggregation of NALUs. When using all tools combined, superior performance can be achieved, and high-quality compressed video over best-effort IP may finally become a reality.

Acknowledgment

The author would like to thank the many implementers of the JM codec software, which created a remarkably stable and efficient tool for video coding research. He would also like to thank VCEG and JVT for the collaborative work and the technically outstanding discussions and contributions. The author is proud to be able to call himself a member of these groups.

References

[1] T. Wiegand, G. J. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 560–576, July 2003.
[2] T. Stockhammer, M. Hannuksela, and T. Wiegand, "H.264/AVC in wireless environments," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 657–673, July 2003.
[3] J. Postel and J. K. Reynolds, "File Transfer Protocol," RFC959, 1985.
[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, "Hypertext Transfer Protocol-HTTP/1.1," RFC 2616, 1999.
[5] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini, and G. J. Sullivan, "Rate-Constrained Coder Control and Comparison of Video Coding Standards," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, pp. 688–703, July 2003.
[6] J. Postel, "Transmission Control Protocol," RFC 793, 1981.
[7] S. Floyd, "Congestion Control Principles," RFC2914, BCP41, 2000.
[8] S. Floyd *et al.*, "Equation-based congestion control for unicast applications," in *SIGCOMM 2000*, Stockholm, Sweden, Aug. 2000, pp. 43–56.
[9] J. Postel, "Internet Protocol," RFC 791, 1981.
[10] ——, "User Datagram Protocol," RFC 768, 1980.
[11] D. D. Clark and D. L. Tennenhouse, "Architectural considerations for a new generation of protocols," in *SIGCOMM Symp. Communications Architectures and Protocols*, Philadelphia, PA, Sept. 1990, pp. 200–208.
[12] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, 1996.
[13] H. Schulzrinne, "RTP Profile for Audio and Video Conferences With Minimal Control," RFC 1890, 1996.
[14] D. Wing, B. Thompson, and T. Koren, "Tunneling multiplexed compressed RTP ('TCRTP')," Internet Draft, Work in Progress, Nov. 2002.
[15] J. Rosenberg and H. Schulzrinne, "An RTP Payload Format for Generic Forward Error Correction," RFC 2733, 1999.
[16] C. Perkins, I. Kouvelas, O. Hodson, V. Hardman, M. Handley, J. C. Bolot, A. Vega-Garcia, and S. Fosse-Parisis, "RTP Payload for Redundant Audio Data," RFC2198, 1997.
[17] *Call Signalling Protocols and Media Stream Packetization for Packet-Based Multimedia Communication Systems*, ITU-T Recommendation H.225.0, 1998.
[18] M. Handley, H. Schulzrinne, E. Schooler, and J. Rosenberg, "SIP: Session Initiation Protocol," RFC 2543, 1999.
[19] M. Handley and V. Jacobson, "SDP: Session Description Protocol," RFC 2326, 1998.
[20] H. Schulzrinne, A. Rao, and R. Lanphier, "Real Time Streaming Protocol (RTSP)," RFC 2326, 1998.
[21] *Information Technology—Generic Coding of Moving Pictures and Associated Audio Information: Systems*, ISO/IEC 13 818-1:2000.
[22] S. Wenger and T. Stockhammer. (2001) H.264 Over IP and H.324 Framework. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/0109_San/VCEG-N52.doc
[23] Y. Wang and Q. Zhu, "Error control and concealment for video communication: A review," *Proc. IEEE*, vol. 86, pp. 974–997, May 1998.
[24] Y. Wang, G. Wen, S. Wenger, and A. K. Katsaggelos, "Review of error resilient techniques for video communications," *IEEE Signal Processing Mag.*, vol. 17, pp. 61–82, July 2000.
[25] *Video Coding for Low Bitrate Communication, Version 2*, ITU-T Recommendation H.263, 1998.
[26] G. Côté and F. Kossentini, "Optimal intra coding of blocks for robust video communication over the internet," *EUROSIP J. Image Commun.—Special Issue on Real-time Video Over the Internet*, pp. 25–34, Sept. 1999.
[27] S. Wenger, T. Stockhammer, and M. Hannuksela, "RTP payload format for JVT video," Internet Draft, Work in Progress, draft-wenger-avt-rtp-jvt-00.txt, Mar. 2003.
[28] S. Fukunaga, T. Nakai, and H. Inoue, "Error resilient video coding by dynamic replacing of reference pictures," in *IEEE Global Telecommunications Conf.*, vol. 3, New York, Nov. 1996, pp. 1503–1508.
[29] S. Wenger, "Video redundancy coding in $\mathrm{H.263+}$," in *Proc. AVSPN 97*, Aberdeen, U.K., 1997.
[30] S. Wenger and M. Horowitz. (2002) Scattered Slices: A New Error Resilience Tool for H.264. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/0202_Gen/JVT-B027.doc
[31] T. Stockhammer, T. Wiegand, T. Oelbaum, and F. Obermeier, "Video coding and transport layer techniques for H.264-based transmission over packet-lossy networks," in *Proc. ICIP 2003*, Barcelona, Spain, to be published.
[32] T. Stockhammer and S. Wenger, "Standard-Compliant enhancement of JVT coded video for transmission over fixed and wireless IP," in *Proc. IWDC 2002*, Capri, Italy, Sept. 2002.
[33] M. M. Hannuksela, Y.-K. Wang, and M. Gabbouj, "Sub-picture: ROI coding and unequal error protection," presented at the IEEE 2002 Int. Conf. Image Processing (ICIP'2002), Rochester, NY, Sept. 2002.
[34] T. Stockhammer, T. Oelbaum, T. Wiegand, and D. Marpe. (2001) H.264 Simulation Results for Common Conditions for H.323/Internet Case. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/0109_San/VCEG-N50.doc
[35] H.264/AVC Codec Software Archive.. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/reference_software
[36] S. Wenger and M. Horowitz. (2002) Scattered Slices: Simulation Results. [Online]ftp://ftp.imtc-files.org/jvt-experts/2002_05_Fairfax/JVT-C090.doc
[37] S. Wenger. (2001) Common Conditions for Wire-Line, Low Delay IP/UDP/RTP Packet Loss Resilient Testing. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/0109_San/VCEG-N79r1.doc
[38] VCEG. (1999) Internet Error Patterns VCEG-O38r1.doc. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/9910_Red/Q15-I16r1.zip
[39] JVT document archives.. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts and ftp://ftp.imtc-files.org/jvt-experts
[40] A. Joch. (2001) Initial Results From a Near-Real-Time H.264 Encoder. VCEG-O38r1.doc. [Online]. Available: ftp://ftp.imtc-files.org/jvt-experts/0112_Pat/VCEG-O38r1.doc

Stephan Wenger received the diploma and Dr.-Ing degrees in computer science from Technische Universität Berlin (TU Berlin), Berlin, Germany, in 1989 and 1995, respectively.

He is with the Communications and Operating Systems Group in the Computer Science Department, TU Berlin. His professional work includes positions such as a Project Manager at TELES AG, Berlin, Germany, and Technical Advisor and Consultant for several companies including Polycom, Siemens, Microsoft, and Intel. He has also helped start several companies in the field of multimedia coding, and serves on the Board of Directors of UB Video Inc., a leading supplier of video compression software. Recently, he has joined TeleSuite as their Chief Video Scientist. His research interests are both in protocol design and media coding for multimedia systems. He has authored or co-authored several journals and conference publications, important standardization contributions, Internet RFCs, and book chapters. He has held several seminars on technical and marketing topics of multimedia communication. He is also very active in the standardization process for new multimedia technologies, especially in the IETF and the ITU-T, where he chairs *ad-hoc* committees in Q.6/16 and in the Joint Video Team composed of video experts from MPEG and Q.6/16. He currently holds two international patents with several pending