ELSEVIER

Signal Processing: *Image Communication* 15 (1999) 95–126

SIGNAL PROCESSING: IMAGE COMMUNICATION

www.elsevier.nl/locate/image

Scalable Internet video using MPEG-4

Hayder Radha*, Yingwei Chen, Kavitha Parthasarathy, Robert Cohen

Philips Research, 345 Scarborough Rd, Briarcliff Manor, New York, 10510, USA

Abstract

Real-time streaming of audio-visual content over Internet Protocol (IP) based networks has enabled a wide range of multimedia applications. An Internet streaming solution has to provide real-time delivery and presentation of a continuous media content while compensating for the lack of Quality-of-Service (QoS) guarantees over the Internet. Due to the variation and unpredictability of bandwidth and other performance parameters (e.g. packet loss rate) over IP networks, in general, most of the proposed streaming solutions are based on some type of a data loss handling method and a layered video coding scheme. In this paper, we describe a real-time streaming solution suitable for non-delay-sensitive video applications such as video-on-demand and live TV viewing.

The main aspects of our proposed streaming solution are:

1. An MPEG-4 based scalable video coding method using both a prediction-based base layer and a fine-granular enhancement layer;

2. An integrated transport-decoder buffer model with priority re-transmission for the recovery of lost packets, and continuous decoding and presentation of video.

In addition to describing the above two aspects of our system, we also give an overview of a recent activity within MPEG-4 video on the development of a fine-granular-scalability coding tool for streaming applications. Results for the performance of our scalable video coding scheme and the re-transmission mechanism are also presented. The latter results are based on actual testing conducted over Internet sessions used for streaming MPEG-4 video in real-time.

1. Introduction

Real-time streaming of multimedia content over Internet Protocol (IP) networks has evolved as one of the major technology areas in recent years. A wide range of interactive and non-interactive multimedia Internet applications, such as news on-demand, live TV viewing, and video conferencing rely on end-to-end streaming solutions. In general, streaming solutions are required to maintain real-time delivery and presentation of the multimedia content while compensating for the lack of Quality-of-Service (QoS) guarantees over IP networks. Therefore, any Internet streaming system has to take into consideration key network performance parameters such as bandwidth, end-to-end delay, delay variation, and packet loss rate.

To compensate for the unpredictability and variability in bandwidth between the sender and receiver(s) over the Internet and Intranet networks, many streaming solutions have resorted to variations of layered (or scalable) video coding methods (see for example [22,24,25]). These

* Corresponding author.

E-mail address: hmr@philabs.research.philips.com (H. Radha)

0923-5965/99/$ - see front matter
PII: S0923-5965(99)00026-0

solutions are typically complemented by packet loss recovery [22] and/or error resilience mechanisms [25] to compensate for the relatively high packet-loss rate usually encountered over the Internet [2,30,32,33,35,47].

Most of the references cited above and the majority of related modeling and analytical research studies published in the literature have focused on delay-sensitive (point-to-multipoint or multipoint-to-multipoint) applications such as video conferencing over the Internet Multicast Backbone – MBone. When compared with other types of applications (e.g. entertainment over the Web), these delay-sensitive applications impose different kind of constraints, such as low encoder complexity and very low end-to-end delay. Meanwhile, entertainment-oriented Internet applications such as news and sports on-demand, movie previews and even 'live' TV viewing represent a major (and growing) element of the real-time multimedia experience over the global Internet [9].

Moreover, many of the proposed streaming solutions are based on either proprietary or video coding standards that were developed at times prior to the phenomenal growth of the Internet. However, under the audio, visual, and system activities of the ISO MPEG-4 work, many aspects of the Internet have being taken into consideration when developing the different parts of the standard. In particular, a recent activity in MPEG-4 video has focused on the development of a scalable compression tool for streaming over IP networks [4,5].

In this paper, we describe a real-time streaming system suitable for non-delay-sensitive[1] video applications (e.g. video-on-demand and live TV viewing) based on the MPEG-4 video-coding standard. The main aspects of our real-time streaming system are:

1. A layered video coding method using both a prediction-based base layer and a fine-granular enhancement layer: This solution follows the recent development in the MPEG-4 video group for the standardization of a scalable video compression tool for Internet streaming applications [3,4,6].
2. An integrated transport-decoder buffer model with a re-transmission based scheme for the recovery of lost packets, and continuous decoding and presentation of video.

The remainder of this paper is organized as follows. In Section 2 we provide an overview of key design issues one needs to consider for real-time, non-delay-sensitive IP streaming solutions. We will also highlight how our proposed approach addresses these issues. Section 3 describes our real-time streaming system and its high level architecture. Section 4 details the MPEG-4 based scalable video coding scheme used by the system, and provides an overview of the MPEG-4 activity on fine-granular-scalability. Simulation results for our scalable video compression solution are also presented in Section 4. In Section 5, we introduce the integrated transport layer-video decoder buffer model with re-transmission. We also evaluate the effectiveness of the re-transmission scheme based on actual tests conducted over the Internet involving real-time streaming of MPEG-4 video.

2. Design considerations for real-time streaming

The following are some high-level issues that should be considered when designing a real-time streaming system for entertainment oriented applications.

2.1. System scalability

The wide range of variation in effective bandwidth and other network performance characteristics over the Internet [33,47] makes it necessary to pursue a scalable streaming solution. The variation in QoS measures (e.g. effective bandwidth) is not only present across the different access technologies to the Internet (e.g. analog modem, ISDN, cable modem, LAN, etc.), but it can even be observed over relatively short periods of time over a particular session [8,33]. For example, a recent study shows that the effective bandwidth

[1] Delay sensitive applications are normally constrained by an end-to-end delay of about 300–500 ms. Real-time, non-delay-sensitive applications can typically tolerate a delay on the order of few seconds.

of a cable modem access link to the Internet may vary between 100 kbps to 1 Mbps [8]. Therefore, any video-coding method and associated streaming solution has to take into consideration this wide range of performance characteristics over IP networks.

2.2. *Video compression complexity, scalability, and coding efficiency*

The video content used for on-demand applications is typically compressed off-line and stored for later viewing through unicast IP sessions. This observation has two implications. First, the complexity of the video encoder is not as major an issue as in the case with interactive multipoint-to-multipoint or even point-to-point applications (e.g. video conferencing and video telephony) where compression has to be supported by every terminal. Second, since the content is not being compressed in real-time, the encoder cannot employ a variable-bit-rate (VBR) method to adapt to the available bandwidth. This emphasizes the need for coding the material using a scalable approach. In addition, for multicast or unicast applications involving a large number of point-to-multipoint sessions, only one encoder (or possibly very few encoders for simulcast) is (are) usually used. This observation also leads to a relaxed constraint on the complexity of the encoder, and highlights the need for video scalability. As a consequence of the relaxed video-complexity constraint for entertainment-oriented IP streaming, there is no need to *totally* avoid such techniques as motion estimation which can provide a great deal of coding efficiency when compared with replenishment-based solutions [24].

Although it is desirable to generate a scalable video stream for a wide range of bit-rates (e.g. 15 kbps for analog-modem Internet access to around 1 Mbps for cable-modem/ADSL access), it is virtually impossible to achieve a good coding-efficiency/video-quality tradeoff over such a wide range of rates. Meanwhile, it is equally important to emphasize the impracticality of coding the video content using simulcast compression at multiple bit-rates to cover the same wide range. First, simulcast compression requires the creation of many streams (e.g. at 20, 40, 100, 200, 400, 600, 800 and 1000 kbps). Second, once a particular simulcast bitstream (coded at a given bit-rate, say R) is selected to be streamed over a given Internet session (which initially can accommodate a bit-rate of R or higher), then due to possible wide variation of the available bandwidth over time, the Internet session bandwidth may fall below the bit-rate R. Consequently, this decrease in bandwidth could significantly degrade the video quality. One way of dealing with this issue is to switch, in real-time, among different simulcast streams. This, however, increases complexities on both the server and the client sides, and introduces synchronization issues.

A good practical alternative to this issue is to use video scalability over *few ranges* of bit-rates. For example, one can create a scalable video stream for the analog/ISDN access bit-rates (e.g. to cover 20–100 kbps bandwidth), and another scalable stream for a higher bit-rate range (e.g. 200 kbps–1 Mbps). This approach leads to another important requirement. Since each scalable stream will be build on the top of a video base layer, this approach implies that multiple base layers will be needed (e.g. one at 20 kbps, another at 200 kbps, and possibly another at 1 Mbps). Therefore, it is quite desirable to deploy a video compression standard that provides good coding efficiency over a rather wide range of possible bit-rates (in the above example 20 kbps, 200 kbps and 1 Mbps). In this regard, due to the many video-compression tools provided by MPEG-4 for achieving high coding efficiency and in particular at low bit-rates, MPEG-4 becomes a very attractive choice for compression.

2.3. *Streaming server complexity*

Typically, a unicast server has to output tens, hundreds, or possibly thousands of video streams simultaneously. This greatly limits the type of processing the server can perform on these streams in real-time. For example, although the separation of an MPEG-2 video stream into three temporal layers (I, P and B) is a feasible approach for a scalable multicast (as proposed in [22]), it will be quite difficult to apply the same method to a large

number of unicast streams. This is the case since the proposed layering requires some parsing of the compressed video bitstream. Therefore, it is desirable to use a very simple scalable video stream that can be easily processed and streamed for unicast sessions. Meanwhile, the scalable stream should be easily divisible into multiple streams for multicast IP similar to the receiver-driven paradigm used in [22,24].

Consequently, we adopt a single, fine-granular enhancement layer that satisfies these requirements. This simple scalability approach has two other advantages. First, it requires only a single enhancement layer decoder at the receiver (even if the original fine-granular stream is divided into multiple sub-streams). Second, the impact of packet losses is localized to the particular enhancement-layer picture(s) experiencing the losses. These and other advantages of the proposed scalability approach will become clearer later in the paper.

2.4. Client complexity and client-server communication issues

There is a wide range of clients that can access the Internet and experience a multimedia streaming application. Therefore, a streaming solution should take into consideration a scalable decoding approach that meets different client-complexity requirements. In addition, one has to incorporate robustness into the client for error recovery and handling, keeping in mind key client-server complexity issues. For example, the deployment of an elaborate feedback scheme between the receivers and the sender (e.g. for flow control and error handling) is not desirable due to the potential implosion of messages at the sender [2,34,35]. However, simple re-transmission techniques have been proven effective for many unicast and multicast multimedia applications [2,10,22,34]. Consequently, we employ a re-transmission method for the recovery of lost packets. This method is combined with a client-driven flow control model that ensures the continuous decoding and presentation of video while minimizing the server complexity.

In summary, a real-time streaming system tailored for entertainment IP applications should provide a good balance among these requirements: (a) scalability of the compressed video content, (b) coding efficiency across a wide range of bit-rates, (c) low complexity at the streaming server, and (d) handling of lost packets and end-to-end flow control using a primarily client-driven approach to minimize server complexity and meet overall system scalability requirements. These elements are addressed in our streaming system as explained in the following sections.

3. An overview of the scalable video streaming system

The overall architecture of our scalable video streaming system is shown in Fig. 1.[2] The system consists of three main components: an MPEG-4 based scalable video encoder, a real-time streaming server, and a corresponding real-time streaming client which includes the video decoder.

MPEG-4 is an international standard being developed by the ISO Moving Picture Experts Group for the coding and representation of multimedia content.[3] In addition to providing standardized methods for decoding compressed audio and video, MPEG-4 provides standards for the representation, delivery, synchronization, and interactivity of audiovisual material. The powerful MPEG-4 tools yield good levels of performance at low bit-rates, while at the same time they present a wealth of new functionality [20].

The video encoder generates two bitstreams: a base-layer and an enhancement-layer compressed video. An MPEG-4 compliant stream is coded based on an MPEG-4 video Verification Model (VM).[4] This stream, which represents the base

[2] The figure illustrates the architecture for a single, unicast server-client session. Extending the architecture shown in the figure to multiple unicast sessions, or to a multicast scenario is straightforward.

[3] http://drogo.cselt.stet.it/mpeg/

[4] The VM is a common set of tools that contain detailed encoding and decoding algorithms used as reference for testing new functionalities. The video encoding was based on the MPEG-4 video group, MoMuSys software Version VCD-06-980625.

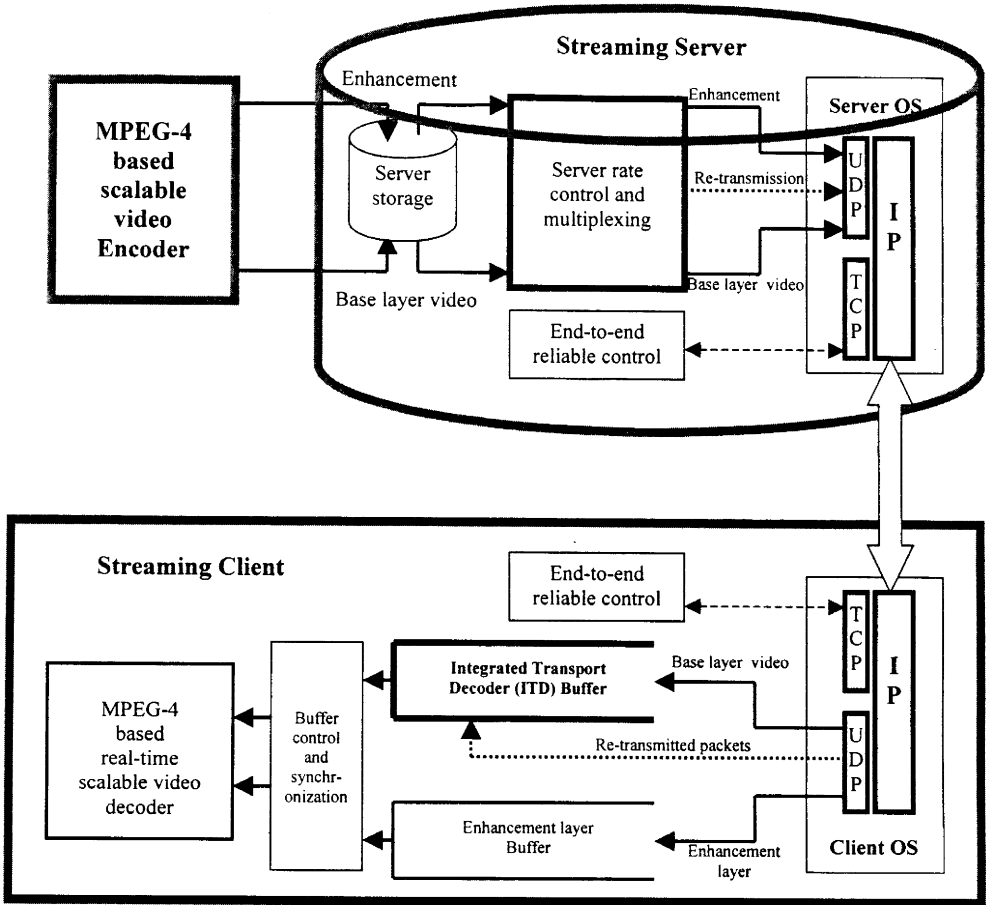

Fig. 1. The end-to-end architecture of an MPEG-4 based scalable video streaming system.

layer of the scalable video encoder output, is coded at a low bit-rate. The particular rate selected depends on the overall range of bit-rates targeted by the system and the complexity of the source material. For example, to serve clients with analog/ISDN modems' Internet access, the base-layer video is coded at around 15–20 kbps. The video enhancement layer is coded using a single fine-granular-scalable bitstream. The method used for coding the enhancement layer follows the recent development in the MPEG-4 video fine-granular-scalability (FGS) activity for Internet streaming applications [4,5]. For the above analog/ISDN-modem access example, the enhancement layer stream is over-coded to a bit-rate around 80–100 kbps. Due to the fine granularity of the enhancement layer, the server can easily select and adapt to the desired bit-rate based on the conditions of the network. The scalable video coding aspects of the system are covered in Section 4.

The server outputs the MPEG-4 base-layer video at a rate that follows very closely the bit-rate at which the stream was originally coded. This aspect of the server is crucial for minimizing underflow and overflow events at the client. Jitter is introduced at the server output due, in part, to the packetization of the compressed video streams. Real-time Transport Protocol (RTP) packetization [15,39] is used to multiplex and synchronize the

base and enhancement layer video. This is accomplished through the time-stamp fields supported in the RTP header. In addition to the base and enhancement streams, the server re-transmits lost packets in response to requests from the client. The three streams (base, enhancement and re-transmission) are sent using the User Datagram Protocol (UDP) over IP. The re-transmission requests between the client and the server are carried in an end-to-end, reliable control session using Transmission Control Protocol (TCP). The server rate-control aspects of the system are covered in Section 5.

In addition to a real-time MPEG-4 based, scalable video decoder, the client includes buffers and a control module to regulate the flow of data and ensure continuous and synchronized decoding of the video content. This is accomplished by deploying an Integrated Transport Decoder (ITD) buffer model which supports packet-loss recovery through re-transmission requests. The ITD buffer model and the corresponding re-transmission method are explained in Section 5.

4. MPEG-4 based scalable video coding for streaming

4.1. Overview of video scalability

Many scalable video-coding approaches have been proposed recently for real-time Internet applications. In [22] a temporal layering scheme is applied to MPEG-2 video coded streams where different picture types (I, P and B) are separated into corresponding layers (I, P and B video layers). These layers are multicasted into separate streams allowing receivers with different session-bandwidth characteristics to subscribe to one or more of these layers. In conjunction with this temporal layering scheme, a re-transmission method is used to recover lost packets. In [25] a spatio-temporal layering scheme is used where temporal compression is based on hierarchical conditional replenishment and spatial compression is based on a hybrid DCT/subband transform coding.

In the scalable video coding system developed in [45], a 3-D subband transform with camera-pan compensation is used to avoid motion compensation drift due to partial reference pictures. Each subband is encoded with progressively decreasing quantization step sizes. The system can support, with a single bitstream, a range of bit-rates from kilobits to megabits and various picture resolutions and frame rates. However, the coding efficiency of the system depends heavily on the type of motion in the video being encoded. If the motion is other than camera panning, then the effectiveness of the temporal redundancy exploitation is limited. In addition, the granularity of the supported bit-rates is fairly coarse.

Several video scalability approaches have been adopted by video compression standards such as MPEG-2, MPEG-4 and H.263. Temporal, spatial and quality (SNR) scalability types have been defined in these standards. All of these types of scalable video consist of a Base Layer (BL) and one or multiple Enhancement Layers (ELs). The BL part of the scalable video stream represents, in general, the minimum amount of data needed for decoding that stream. The EL part of the stream represents additional information, and therefore it enhances the video signal representation when decoded by the receiver.

For each type of video scalability, a certain *scalability structure* is used. The scalability structure defines the relationship among the pictures of the BL and the pictures of the enhancement layer. Fig. 2 illustrates examples of video scalability structures. MPEG-4 also supports object-based scalability structures for arbitrarily shaped video objects [17,18].

Another type of scalability, which has been primarily used for coding still images, is *fine-granular scalability*. Images coded with this type of scalability can be decoded progressively. In other words, the decoder can start decoding and displaying the image after receiving a very small amount of data. As more data is received, the quality of the decoded image is progressively enhanced until the complete information is received, decoded, and displayed. Among lead international standards, progressive image coding is one of the modes supported in JPEG [16] and the still-image, texture coding tool in MPEG-4 video [17].

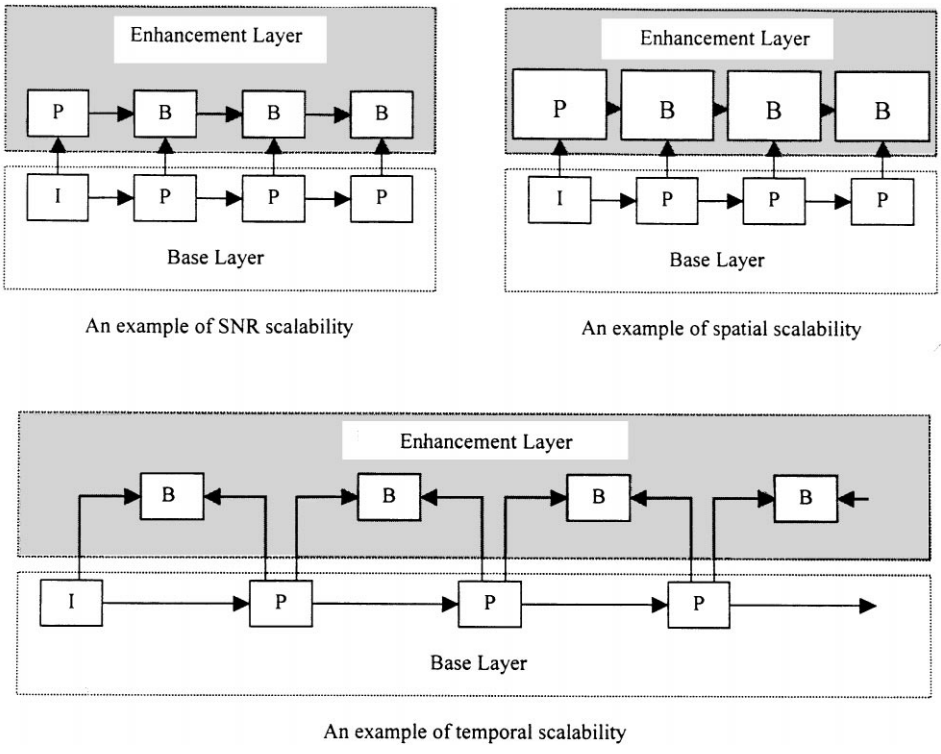

Fig. 2. Examples of video scalability structures.

When compared with non-scalable methods, a disadvantage of video scalable compression is its inferior coding efficiency. In order to increase coding efficiency, video scalable methods normally rely on relatively complex structures (such as the spatial and temporal scalability examples shown in Fig. 2). By using information from as many pictures as possible from both the BL and EL, coding efficiency can be improved when compressing an enhancement-layer picture. However, using prediction among pictures within the enhancement layer either eliminates or significantly reduces the fine-granular scalability feature, which is desirable for environments with a wide range of available bandwidth (e.g. the Internet). On the other hand, using a fine-granular scalable approach (e.g. progressive JPEG or the MPEG-4 still-image coding tool) to compress each picture of a video sequence prevents the employment of prediction among the pictures, and consequently degrades coding efficiency.

4.2. MPEG-4 video based fine-granular-scalability (FGS)

In order to strike a balance between coding-efficiency and fine-granularity requirements, a recent activity in MPEG-4 adopted a hybrid scalability structure characterized by a DCT motion compensated base layer and a fine granular scalable enhancement layer [4,5]. This scalability structure is illustrated in Fig. 3. The video coding scheme used by our system is based on this scalability structure [5]. Under this structure, the server can transmit part or all of the over-coded enhancement layer to the receiver. Therefore, unlike the scalability solutions shown in Fig. 2, the FGS structure enables the streaming system to adapt to varying network conditions. As explained in Section 2, the FGS feature is especially needed when the video is pre-compressed and the condition of the particular session (over which the

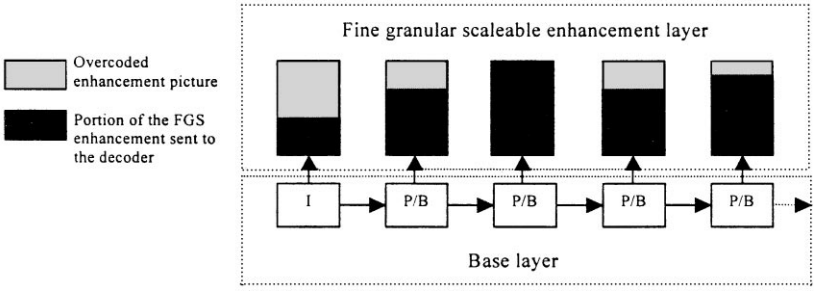

Fig. 3. Video scalability structure with fine-granularity.

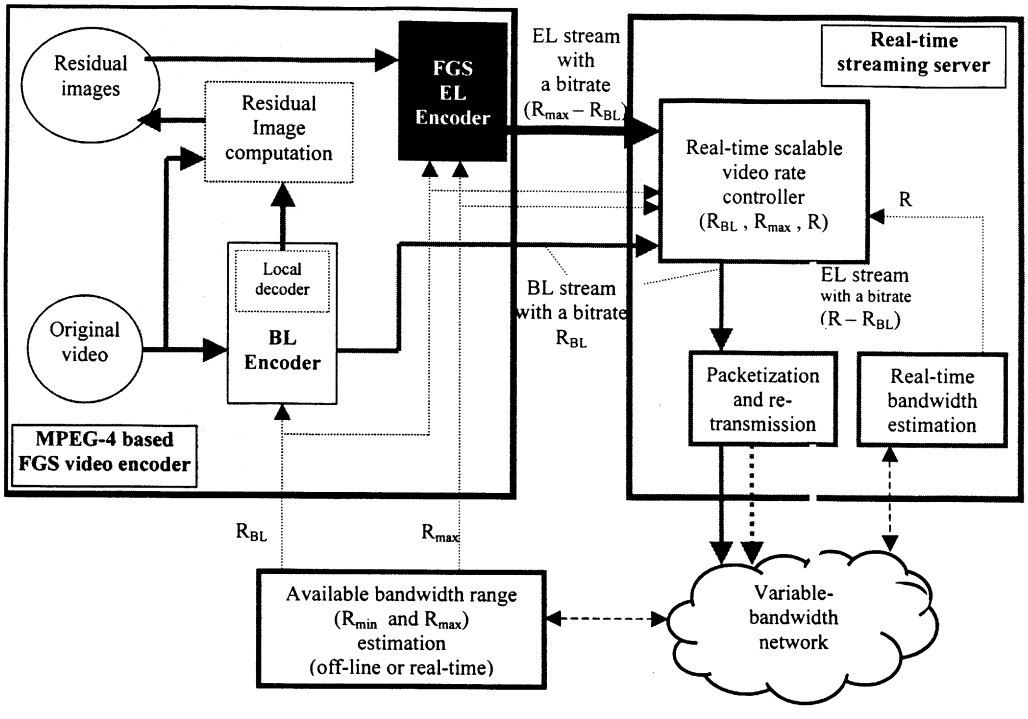

Fig. 4. A streaming system employing the MPEG-4 based fine-granular video scalability.

bitstream will be delivered) is not known at the time when the video is coded.

Fig. 4 shows the internal architecture of the MPEG-4 based FGS video encoder used in our streaming system. The base layer carries a minimally acceptable quality of video to be reliably delivered using a re-transmission, packet-loss recovery method. The enhancement layer improves upon the base layer video, fully utilizing the estimated available bandwidth (Section 5.5). By employing a motion compensated base layer, coding efficiency from temporal redundancy exploitation is partially retained. The base and a single-enhancement layer streams can be either stored for later transmission, or can be directly streamed by the server in real-time. The encoder interfaces with a system module that performs estimates of the range of bandwidth $[R_{min}, R_{max}]$ that can be

supported over the desired network. Based on this information, the module conveys to the encoder the bit-rate $R_{BL} \leqslant R_{min}$ that must be used to compress the base-layer video.[5] The enhancement layer is over-coded using a bit-rate $(R_{max} - R_{BL})$. It is important to note that the range $[R_{min}, R_{max}]$ can be determined off-line for a particular set of Internet access technologies. For example, $R_{min} = 20$ kbps and $R_{max} = 100$ kbps can be used for analogue-modem/ISDN access technologies. More sophisticated techniques can also be employed in real-time to estimate the range $[R_{min}, R_{max}]$. For unicast streaming, an estimate for the available bandwidth R can be generated in real-time for a particular session. Based on this estimate, the server transmits the enhancement layer using a bit-rate R_{EL}:

$$R_{EL} = \min(R_{max} - R_{BL}, R - R_{BL}).$$

Due to the fine granularity of the enhancement layer, its real-time rate control aspect can be implemented with minimal processing (Section 5.5). For multicast streaming, a set of intermediate bit-rates $R_1, R_2, \ldots, R_N$ can be used to partition the enhancement layer into substreams. In this case, N fine-granular streams are multicasted using the bit-rates:

$$R_{e1} = R_1 - R_{BL},$$

$$R_{e2} = R_2 - R_1, \ldots, R_{eN} = R_N - R_{N-1},$$

where

$$R_{BL} < R_1 < R_2 \cdots < R_{N-1} < R_N \leqslant R_{max}.$$

Using a receiver-driven paradigm [24], the client can subscribe to the base layer and one or more of the enhancement layers' streams. As explained earlier, one of the advantages of the FGS approach is that the EL sub-streams can be combined at the receiver into a single stream and decoded using a single EL decoder.

[5] Typically, the base layer encoder will compress the signal using the minimum bit-rate R_{min}. This is especially the case when the BL encoding takes place off-line prior to the time of transmitting the video signal.

There are many alternative compression methods one can choose from when coding the BL and EL layers of the FGS structure shown in Fig. 3. MPEG-4 is highly anticipated to be the next widely-deployed audio-visual standard for interactive multimedia applications. In particular, MPEG-4 video provides superior low-bit-rate coding performance when compared with other MPEG standards (i.e. MPEG-1 and MPEG-2), and provides object-based functionality. In addition, MPEG-4 video has demonstrated its coding efficiency even for medium-to-high bit-rates. Therefore, we use the DCT-based MPEG-4 video tools for coding the base layer. There are many excellent documents and papers describe the MPEG-4 video coding tools [17,18,43,44].

For the EL encoder shown in Fig. 4, any embedded or fine-granular compression scheme can be used. Wavelet-based solutions have shown excellent coding-efficiency and fine-granularity performance for image compression [41,37]. In the following sub-section, we will discuss our wavelet solution for coding the EL of the MPEG-4 based scalable video encoder. Simulation results of our MPEG-4 based FGS coding method will be presented in Section 4.3.2.

4.3. The FGS enhancement layer encoder using wavelet

In addition to achieving a good balance between coding efficiency and fine granularity, there are other criteria that need to be considered when selecting the enhancement layer coding scheme. These criteria include complexity, maturity and acceptability of that scheme by the technical and industrial communities for broad adaptation. The complexity of such scheme should be sufficiently low, in particular, for the decoder. The technique should be reasonably mature and stable. Moreover, it is desirable that the selected technique has some roots in MPEG or other standardization bodies to facilitate its broad acceptability.

Embedded wavelet coding satisfies all of the above criteria. It has proven very efficient in coding still images [38,41] and is also efficient in coding video signals [46]. It naturally provides fine granular scalability, which has always been one of its

strengths when compared to other transform-based coding schemes. Because wavelet-based image compression has been studied for many years now, and because its relationship with sub-band coding is well established there exist fast algorithms and implementations to reduce its complexity. Moreover, MPEG-4 video includes a still-image compression tool based on the wavelet transform [17]. This still-image coding tool supports three compression modes, one of which is fine granular. In addition, the image-compression methods currently competing under the JPEG-2000 standardization activities are based on the wavelet transform. All of the above factors make wavelet based coding for the FGS enhancement layer a very attractive choice.

Ever since the introduction of EZW (Embedded Zerotrees of Wavelet coefficients) by Shapiro [41], much research has been directed toward efficient progressive encoding of images and video using wavelets. Progress in this area has culminated recently with the SPIHT (Set Partitioning In Hierarchical Trees) algorithm developed by Said and Pearlman [38]. The still-image, texture coding tool in MPEG-4 also represents a variation of EZW and gives comparable performance to that of SPIHT.

Compression results and proposals for using different variations of the EZW algorithm have been recently submitted to the MPEG-4 activity on FGS video [6,17,19,40]. These EZW-based proposals include the scalable video coding solution used in our streaming system. Below, we give a brief overview of the original EZW method and highlight how the recent wavelet-based MPEG-4 proposals (for coding the FGS EL video) differ from the original EZW algorithm. Simulation results are shown at the end of the section.

4.3.1. EZW-based coding of the enhancement-layer video

The different variations of the EZW approach [6,17,19,37,38,40,41] are based on: (a) computing a wavelet-transform of the image, and (b) coding the resulting transform by partitioning the wavelet coefficients into sets of hierarchical, *spatial-orientation trees*. An example of a spatial-orientation tree is shown in Fig. 5. In the original EZW algorithm

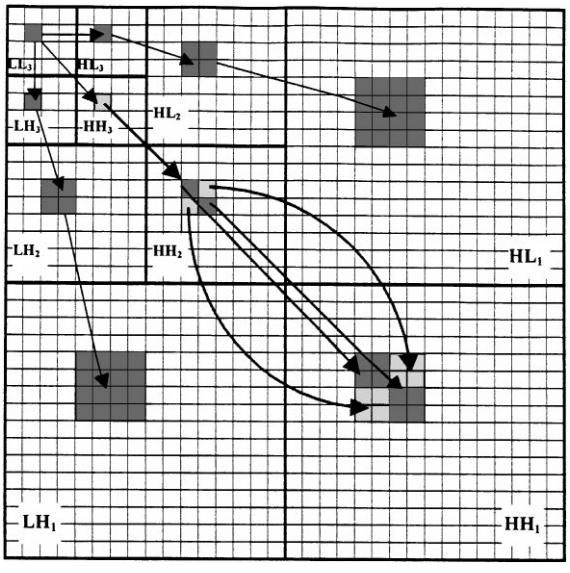

Fig. 5. Examples of the hierarchical, spatial-orientation trees of the zero-tree algorithm.

[41], each tree is rooted at the highest level (most coarse sub-band) of the multi-layer wavelet transform. If there are m layers of sub-bands in the hierarchical wavelet transform representation of the image, then the roots of the trees are in the LL_m sub-band of the hierarchy as shown in Fig. 5. If the number of coefficients in sub-band LL_m is N_m, then there are N_m spatial-orientation trees representing the wavelet transform of the image.

In EZW, coding efficiency is achieved based on the hypothesis of "decaying spectrum": the energies of the wavelet coefficients are expected to decay in the direction from the root of a spatial-orientation tree toward its descendants. Consequently, if the wavelet coefficient c_n of a node n is found insignificant (relative to some threshold $T_k = 2^k$), then it is highly probable that all descendants $D(n)$ of the node n are also insignificant (relative to the same threshold T_k). If the root of a tree and all of its descendants are insignificant then this tree is referred to as a Zero-Tree Root (ZTR). If a node n is insignificant (i.e. $|c_n| < T_k$) but one (or more) of its descendants is (are) significant then this scenario represents a violation of the 'decaying spectrum' hypothesis. Such a node is referred to as an Isolated Zero-Tree (IZT). In the original EZW

algorithm, a significant coefficient c_n (i.e. $|c_n| > T_k$) is coded either positive (POS) or negative (NEG) depending on the sign of the coefficient. Therefore, if $S(n, T_k)$ represents the *significance symbol* used for coding a node n relative to a threshold $T_k = 2^k$, then

$$S(n, T_k) = \begin{cases} \text{ZTR} & \text{if } |c_n| < T_k \text{ and } \max\limits_{m \in D(n)} (|c_m|) < T_k, \\ \text{IZT} & \text{if } |c_n| < T_k \text{ and } \max\limits_{m \in D(n)} (|c_m|) \geqslant T_k, \\ \text{POS} & \text{if } |c_n| \geqslant T_k \text{ and } c_n > 0, \\ \text{NEG} & \text{if } |c_n| \geqslant T_k \text{ and } c_n < 0. \end{cases} \quad (1)$$

There are two main stages (or 'passes') in EZW-based coding algorithms: a dominant pass and a subordinate pass. The execution of a subordinate pass begins after the completion of a dominant pass. Each pass scans a corresponding list of coefficients (dominant list and subordinate list). In the dominant pass, coefficients are scanned in such a way such that a coefficient in a given sub-band is scanned prior to all coefficients belonging to a finer (higher resolution) sub-band. An example of this scanning is shown in Fig. 6. While being scanned, the coefficients are examined for their significance with respect to a threshold $T_k = 2^k$, $k = 0, 1, \ldots, K$, where

$$K = \lfloor \log_2(\max|c_n|) \rfloor.$$

For each threshold value T_k, the EZW algorithm scans and examines the wavelet coefficients for their significance starting with the largest threshold T_K, then T_{K-1}, and so on. Therefore, in all there could be as many as $K + 1$ dominant passes, each of which is followed by a subordinate pass. Due to its embedded nature, the algorithm can stop at any point (within a dominant or subordinate pass) if a certain bit-budget constraint or distortion-measure criterion is achieved. Prior to the execution of the dominant/subordinate-passes stage, the EZW algorithm requires a simple initialization step for computing and transmitting the parameter K, and for initializing the dominant and subordinate lists. A high-level structure of the EZW algorithm is shown in Fig. 7.

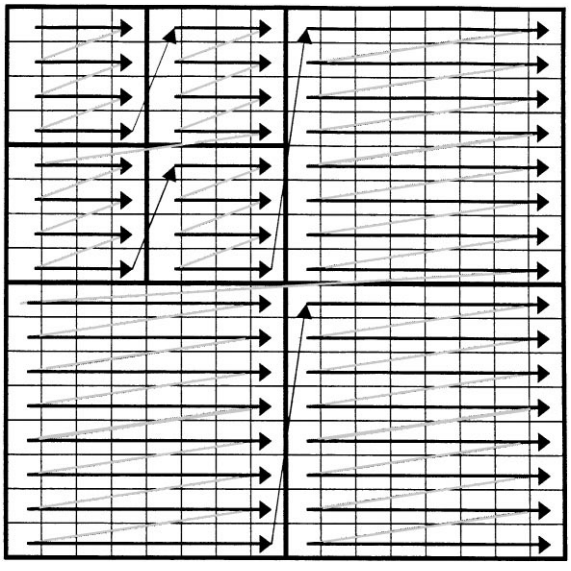

Fig. 6. A sub-band by sub-band scanning order of the EZW algorithm. This is one of the scanning orders supported by the MPEG-4 still-image wavelet coding tool.

Under each dominant pass, and for each scanned coefficient, one of the four above symbols (ZTR, IZT, POS, NEG) is transmitted to the decoder. If a coefficient is declared a zero-tree (ZTR), all of its descendants are not examined, and consequently no extra symbols are needed to code the rest of this tree under the current dominant pass. However, a zero-tree node (and its descendants) has to be examined under subsequent dominant passes relative to smaller thresholds. If a coefficient is POS or NEG, it is removed from the dominant list[6] and put into the subordinate list for further processing by the subordinate pass. This is done since once a coefficient is found significant (POS or NEG), it will also be significant relative to subsequent (smaller) thresholds. If a node n is found to be an isolated zero-tree (IZT), this indicates that further scanning of this node's descendants is needed to identify the significant coefficients under n. At the end of each dominant pass, only zero-tree and isolated zero-tree nodes remain part of the dominant list for

[6] In the original EZW algorithm, the significant coefficients in the wavelet transform are actually set to zero.

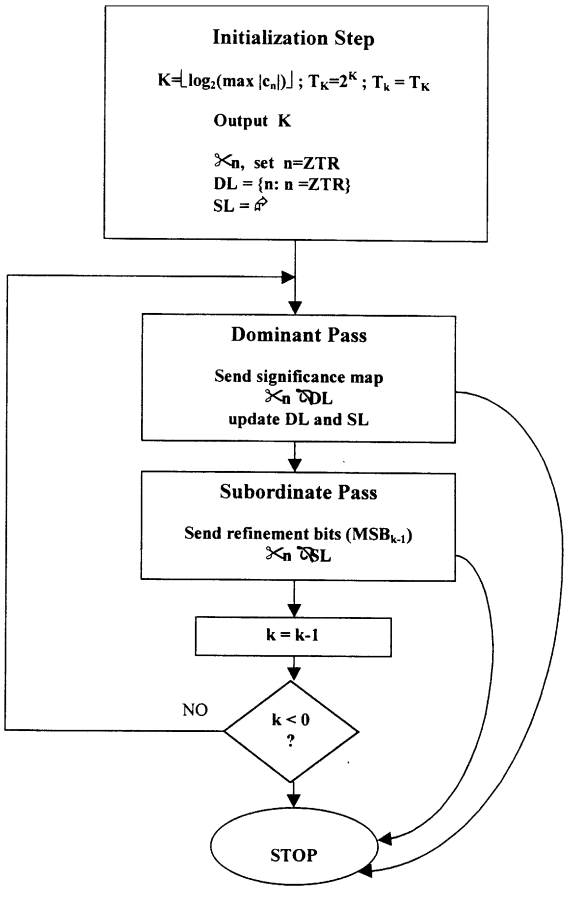

Fig. 7. A high-level structure of the EZW algorithm.

further examination under the next dominant pass. The coding procedure used under a dominant pass of the EZW algorithm is referred to as *significance map* coding.

Under a subordinate pass (i.e. after the corresponding dominant pass examination of the coefficients relative to a threshold T_k), the magnitude of each node in the subordinate list (i.e. each significant POS or NEG coefficient) is refined. This is done by transmitting the next most-significant-bit (MSB_{k-1}) of that coefficient to the decoder.

So far, we have described some of the key elements of the original EZW (for more details we refer the reader to [41]). This description will assist in highlighting the variations (around this algorithm) used by the recent wavelet-based experiments conducted in support of the MPEG-4 FGS video-tool effort. These proposals and experiments fall into two categories. One set of experiments was done using the fine-granular mode of the wavelet-based MPEG-4 still-image, texture-coding tool. The other set of experiments employed the SPIHT algorithm.

Under the MPEG-4 still-image wavelet tool, in addition to using zero-tree (ZTR) and isolated zero-tree (IZT) symbols to code the significance map, two other symbols are used: Value Zero-Tree Root (VZTR) and Value (VAL). If $D(n)$ represents the set of descendants of a node n, then the new symbols used for coding the significance of a coefficient c_n (relative to a threshold T_k) can be expressed as follows:

$$S(n, T_k) = \begin{cases} \mathrm{ZTR} & \text{if } |c_n| < T_k, \text{ and } \max\limits_{m \in D(n)} (|c_m|) < T_k, \\ \mathrm{IZT} & \text{if } |c_n| < T_k \text{ and } \max\limits_{m \in D(n)} (|c_m|) \geqslant T_k, \\ \mathrm{VZTR} & \text{if } |c_n| \geqslant T_k \text{ and } \max\limits_{m \in D(n)} (|c_m|) < T_k, \\ \mathrm{VAL} & \text{if } |c_n| \geqslant T_k \text{ and } \max\limits_{m \in D(n)} (|c_m) \geqslant T_k. \end{cases} \quad (2)$$

Since there are no positive or negative symbols used here, the encoder has to send the sign of each significant coefficient. The *bi-level* quantization option represents the fine-granular scalability mode of the MPEG-4 wavelet still-image coding tool. This mode supports the sub-band by sub-band scanning shown in Fig. 6. The significance map symbols, the significant coefficients' signs, and the refinement bits of already-identified significant coefficients are coded using an arithmetic encoder which derives context-based probability models [17]. Another feature supported by the MPEG-4 still-image tool is the coding of the highest-level sub-band (the 'DC' sub-band coefficients) separately from other sub-bands. The DC sub-band coefficients are quantized and coded in a predictive manner using an adaptive arithmetic encoder. This summarizes some of the major differences between the original EZW algorithm and the MPEG-4 wavelet still-image coding tool.

Under the SPIHT algorithm [38], a different method is used for coding the significance map. The algorithm simply uses only two symbols: 1 (significant) and 0 (insignificant). In this case, the

dominant list (used in the original EZW to scan the insignificant nodes) is replaced here with two lists. One list is used to scan and examine the insignificant coefficients individually (i.e. no examination of the descendants of a node – just the node itself). This list is referred to as the List of Insignificant Pixels (LIP). The other list is used to examine the *sets* of insignificant descendants of nodes in the tree (List of Insignificant Sets – LIS). Therefore, each node in the LIS list represents its descendants' coefficients (but not itself). In SPIHT, the insignificance of a node either refers to the insignificance of its own coefficient (if the node is in the LIP list) or the insignificance of its descendants (if the node is in the LIS list). Therefore, if $\aleph$ represents a set of nodes, then the symbols used for coding the significance map can be expressed as:

$$S(\aleph, T_k) = \begin{cases} 1 & \text{if } \max_{m\in\aleph}(|c_m|) \geqslant T_k, \\ 0 & \text{otherwise.} \end{cases}$$

In this case, if $\aleph = \{n\}$ is a single node, then it is examined during the LIP list scanning, and if $\aleph = \{D(n)\}$ is a set of multiple nodes (i.e. the set of descendants of a node n in the tree), then $\aleph$ is examined during the LIS list scanning.

It is important to note that a particular node can be a member of both lists (LIP and LIS). If both the node *n and* its descendants are insignificant (i.e. the equivalence of having a zero-tree in the original EZW), then n will be a member of both the LIP and LIS sets.[7] Consequently, the dominant[8] pass of the original EZW algorithm is replaced with two sub-passes under SPIHT: one sub-pass for scanning the LIP coefficients and the other for scanning the LIS sets. This is illustrated in Fig. 8. Similar to the MPEG-4 still-image coding tool, for every coefficient found significant during the LIP or LIS scanning, its sign is transmitted, and the coefficient is put in a third list (List of Significant Pixels – LSP). The

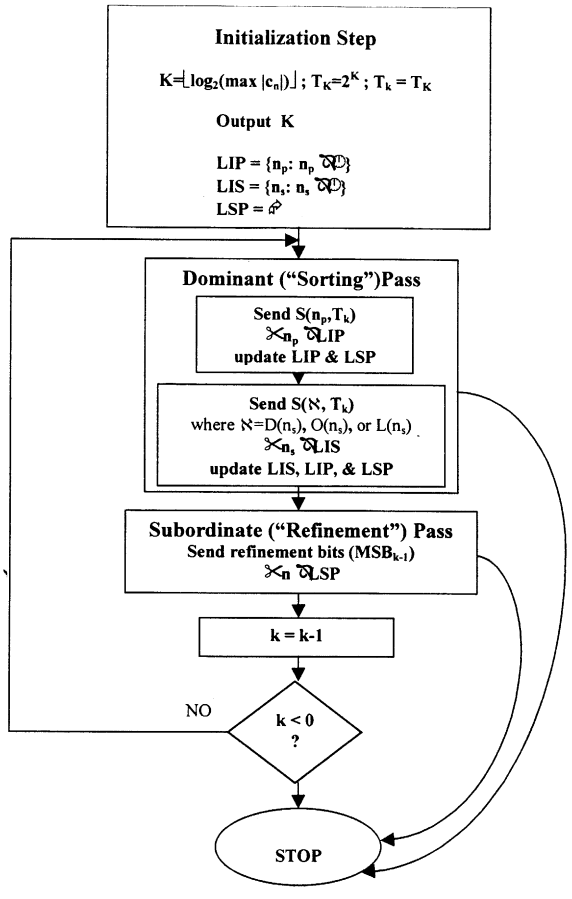

Fig. 8. A simplified structure of the SPIHT algorithm. Here, $O(n)$ is the set of immediate descendants (offsprings) of n, and $L(n)$ are the non-immediate descendants of n. It should be noted that there are more detailed processing and scanning that take place within the dominant pass. This includes the scanning order of the immediate descendants (or offsprings) and non-immediate descendants of a node n_s in the LIS. For more details, the reader is referred to [38].

LSP is used by the subordinate passes (or refinement passes using SPIHT terminology) to send the next MSBs of already identified significant coefficients.

Another distinct feature of the SPIHT algorithm is its hierarchical way of building up its lists. For example, instead of listing all coefficients as members of the dominant list and initializing them to be zero-tree nodes as done in the original EZW algorithm, in SPIHT only the main roots of the

[7] Its membership in the LIS list is basically a pointer to its descendants. However, the node itself does not get examined during the LIS list scanning.

[8] Using SPIHT terminologies, the dominant pass is referred to as the 'sorting pass'.

spatial-orientation trees are put on the LIS and LIP list. These lists are then appended with new nodes deeper in the tree as the dominant ("sorting") passes get executed. Similar to the EZW algorithm, the set of root nodes $\Re$ includes all nodes in the highest-level ('DC') sub-band except the top-left coefficient (the DC coefficient).

This concludes our summary on how the MPEG-4 still-image coding tool and the SPIHT algorithm differ from the original EZW method. As mentioned above, both methods were used to compress residual video signals under the MPEG-4 FGS video experimentation effort. In the next section, we will provide an overview of the MPEG-4 FGS video experiments and show some simulation results.

Before proceeding, it is important to highlight one key point. The EZW-based methods have proven very efficient in encoding still images, due to the high-energy compaction of wavelet coefficients. However, because residual signals possess different statistical properties from those of still images, special care needs to be taken to encode them efficiently. Because the base layer is DCT based, blocking artifacts are observed at low bit-rates. This type of artifacts in the signal will result in unnatural edges and create high-energy wavelet coefficients corresponding to the block edges. Two approaches have been identified to reduce blocking artifacts in the reconstructed images. One is Overlapped Block-matching Motion Compensation, which was used in the scalable wavelet coder developed by [46]. The other is to filter the DCT-coded images, and then compute the residual signals to be refined by the fine granular scalable enhancement layer. This latter approach, which is consistent with the MPEG-4 generic model for scalability [17], is referred to as 'mid-processing'. We will show simulation results in conjunction with and without mid-processing.

4.3.2. Simulation results for the MPEG-4 based fine-granular-scalability coding method

The simulation results presented here are based on the scalability structure shown in Fig. 3. In addition, we use a set of video parameters and test conditions for both the base and enhancement layer as defined by the MPEG-4 activity on FGS video coding [4]. Table 1 shows the MPEG-4 video sequences tested with the corresponding parameters including the base-layer bit-rate. For the enhancement layer, a set of 'bit-rate traces' was defined as shown in Fig. 9. It is important to note, however, that the enhancement layers were over-coded and generated without making any assumptions about these traces. This is to emulate, for example, the scenario when the encoder does not have knowledge of the available bandwidth to be used at a later time for streaming the bitstreams. Another example is the scenario when the encoder is ignorant about the receiver available bandwidth or processing-power capability (even if the video is being compressed in real-time). An enhancement layer trace $t(n)$ identifies the number of bits $e(n)$ that must be used for decoding the nth enhancement layer picture: $e(n) = b(n)*t(n)$, when $b(n)$ is the number of bits used for coding the nth base-layer picture.

Below, we present a summary of the simulation results of using the wavelet coding method based on the SPIHT variation of the EZW algorithm as described in the previous section. (For more details

Table 1
The video sequences and their parameters used in the MPEG-4 FGS experiment

Size	Sequence	Frame rate (fps)	Bitrate (bps)	Quant	Search range
CIF	Foreman	15	124.73k	31	32
	Coastguard		138.34k		32
SIF	Stefan	30	704.29k	15	64
QCIF	Foreman	7.5	26.65k	25	16
	Coastguard		25.76k	20	16

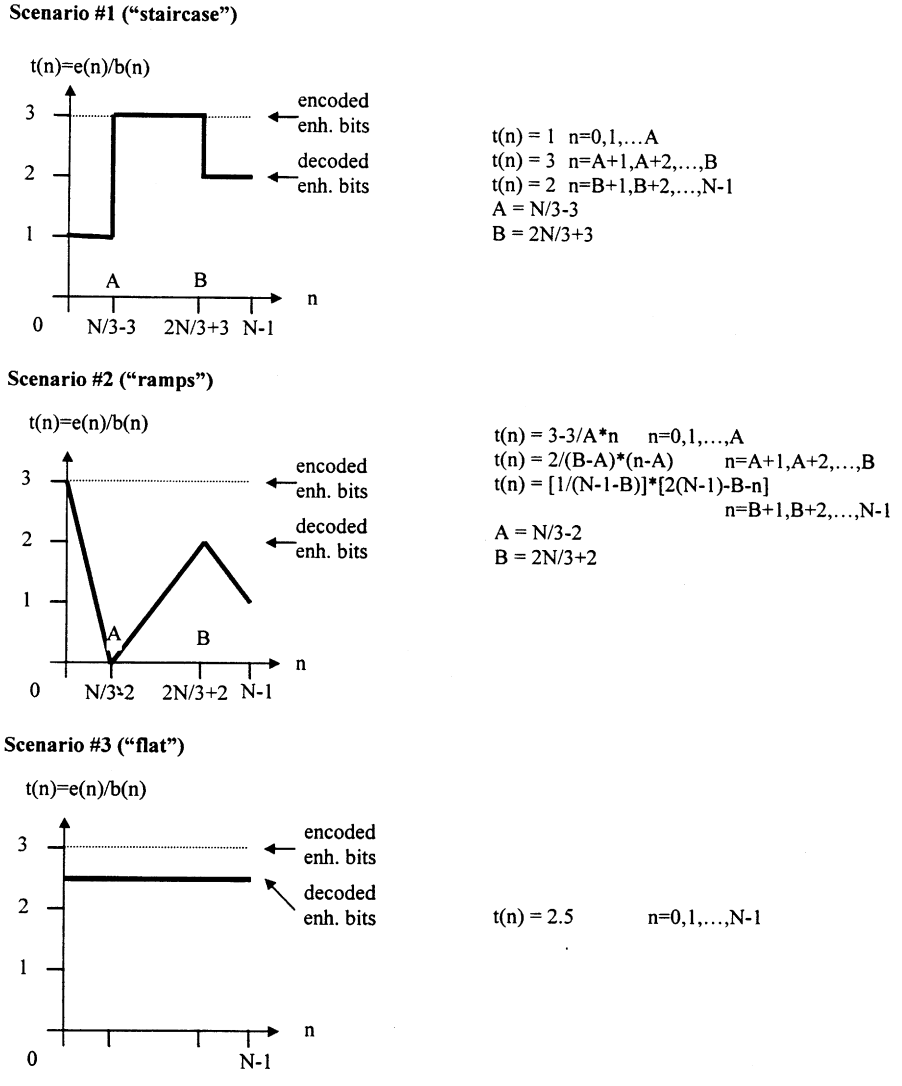

Fig. 9. The bit-rate traces defined by the MPEG-4 FGS video experiment for the enhancement layer.

about the simulation results of using all of the wavelet-based experiments submitted so far to MPEG-4, the reader is referred to [6,19].)

Table 2 shows the average PSNR performance of the wavelet based coding scheme employed in our streaming system using the video sequences and associated testing conditions as defined by the MPEG-4 FGS experiment effort.

Two sets of EL testing scenarios were conducted: one with 'mid-processing' and one without (as explained in the previous section). For each scenario and for each test sequence all of the three bandwidth traces were used. Since our wavelet encoder is bit-wise fine granular, the decoded number of bits per enhancement frame is exactly the same as that of the decoding traces. Therefore, the decoding traces can also be interpreted as the actual number of decoded bits for each enhancement frame. The base layer is encoded using the MPEG-4 MoMuSys software Version VCD-06-980625.

Table 2
Average PSNR performance of the wavelet based coding scheme employed by our system using the video sequences and test parameters defined by the MPEG-4 FGS activity

Sequence	Trace	R_b	SNR_b (Y, U, V)			SNR + (Y, U, V)			SNR + with mid proc		
coastguard_qcif	stair	25.76	27.39	38.77	41.68	1.99	1.87	1.50	2.06	2.00	1.60
	ramp					1.21	1.34	1.06	1.22	1.47	1.08
	flat					2.37	2.39	1.88	2.47	2.58	2.01
foreman_qcif	stair	26.65	28.14	35.29	34.47	3.05	1.97	1.65	3.50	2.79	2.46
	ramp					2.03	1.39	0.97	2.40	2.11	1.66
	flat					3.42	2.26	1.78	3.93	3.06	2.50
coastguard_cif	stair	138.34	26.52	37.65	40.95	1.98	2.36	2.21	1.96	2.65	2.33
	ramp					1.31	1.75	1.72	1.25	2.08	1.87
	flat					2.45	2.33	2.29	2.47	2.59	2.35
foreman_cif	stair	124.73	29.22	34.94	35.43	2.53	2.42	2.34	2.81	2.98	3.04
	ramp					1.76	1.89	1.93	1.98	2.30	2.49
	flat					2.99	2.61	2.53	3.27	3.27	3.37
stefan_sif	stair	704.29	28.36	34.15	33.78	3.35	3.29	2.86	3.53	3.44	3.10
	ramp					2.19	2.00	1.98	2.30	2.14	2.13
	flat					4.24	3.73	3.19	4.46	3.94	3.50

R_b: base layer bit rate in kbits/s
SNR_b: base PSNR (dB)
SNR +: enhancement PSNR improvement
SNR + with mid proc: base layer is deblocked and deringed before residual is computed

A video sequence structure of IPPPP … with one intra picture at the beginning of each sequence is used for these results.

It is clear from the table that, in addition to the fine-granularity benefit of the embedded wavelet coding scheme, the overall video quality (based on the average PSNR measure) is improved by as much as 4.46 dB relative to the base-layer (the Stefan sequence with mid-processing and enhancement bit-rate of 2.5 times the base-layer bit-rate). And as expected, by comparing the PSNR numbers for the cases with and without 'mid-processing', it is clear that employing the deblocking filter on the base-layer prior to computing the residual pictures provides an improvement in the average PSNR values. Increase in PSNR values due to mid-processing is as high as about 15% for the case of hard-to-code sequences at low-bit-rates (e.g. the forman QCIF sequence at 25 kbps). As expected, for relatively easy-to-code sequences or for higher bit-rate scenarios the advantage of mid-processing was not as significant as the previous case (e.g. only about 5% increase in PSNR averages for the 'Stefan' sequence.)

Fig. 10 shows an example of a picture of the 'coastguard' (QCIF) sequence decoded from the base-layer and the corresponding enhanced pictures using decoded frames from the FGS wavelet bitstream. The graceful improvement in the visual quality can be observed from the figure. The base-layer bit-rate used here is $R = 24$ kbps. The enhancement layer bit-rates used are R, $2R$, $3R$, $4R$ and $5R$. It is important to note that the enhancement pictures were derived from a single wavelet stream in a fine-granular manner. Fig. 11 shows the plots of the PSNR values for the same ('coastguard') sequence as a function of the picture number. The plot with the lowest PSNR numbers illustrates the performance of the base-layer coded at $R = 24$ kbps. The plots with the higher PSNR values are for the enhanced sequences decoded using enhancement bit-rates of R, $2R$, $3R$, $4R$ and $5R$, in an ascending order. As noted before, only a single enhancement-layer, wavelet stream was generated.

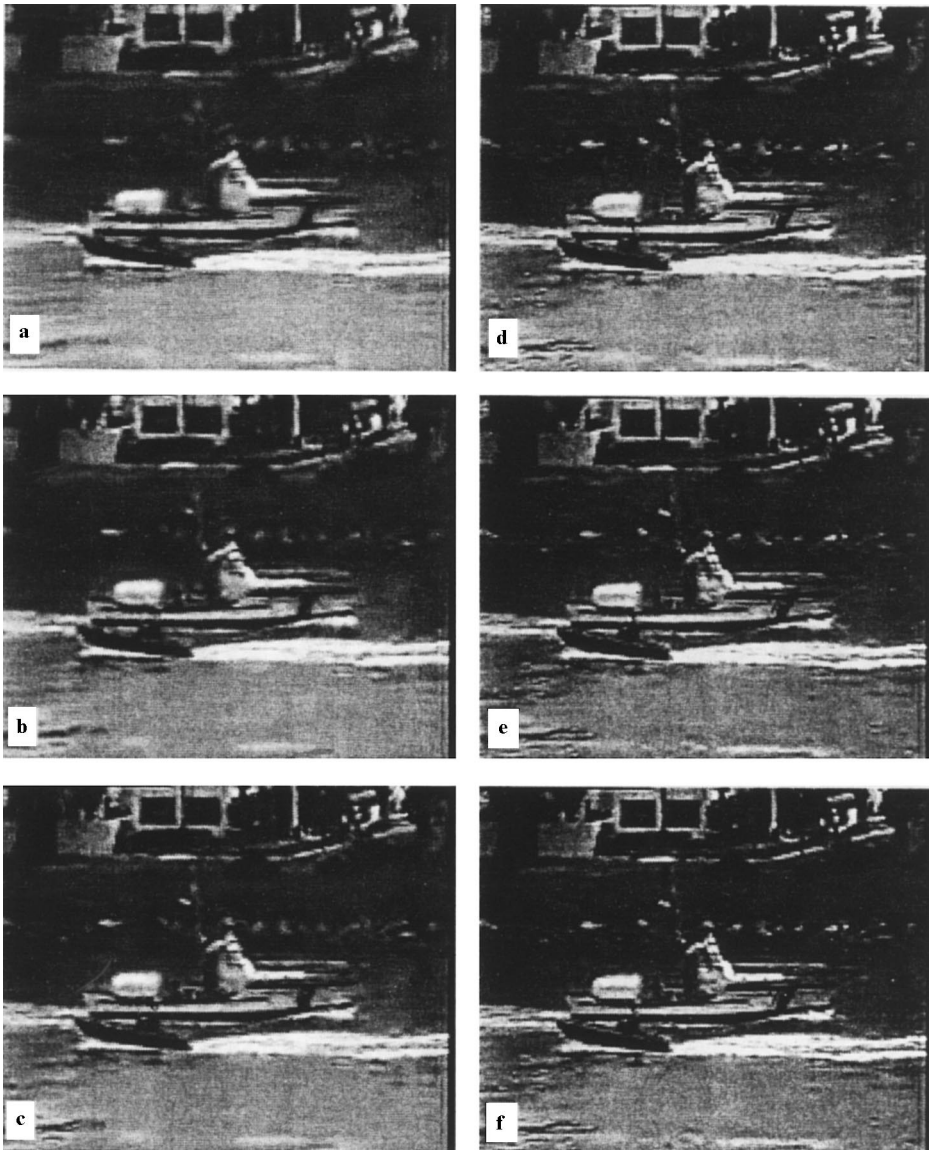

Fig. 10. A picture with different number of bit-rates used for decoding the enhancement layer form the QCIF 'coastguard' sequence. (a) The picture from the base-layer which is coded at a bit-rate $R = 24$ kbps; (b), (c), (d), (e) and (f) are the corresponding pictures decoded using enhancement-layer bit-rates of R, $2R$, $3R$, $4R$ and $5R$, respectively. It is important to note that only a single enhancement-layer wavelet stream was generated, and therefore all of the enhancement pictures were decoded from the same stream in a fine-granular way.

Therefore, all of the enhancement pictures were decoded from the same stream in a fine-granular way. The results shown in the figure were generated without using the deblocking filter on the base-layer frames.

4.4. Concluding remarks on FGS video coding

Standardization of an FGS video method is crucial for a wide deployment of this functionality for Internet users. Therefore, the MPEG-4 FGS video

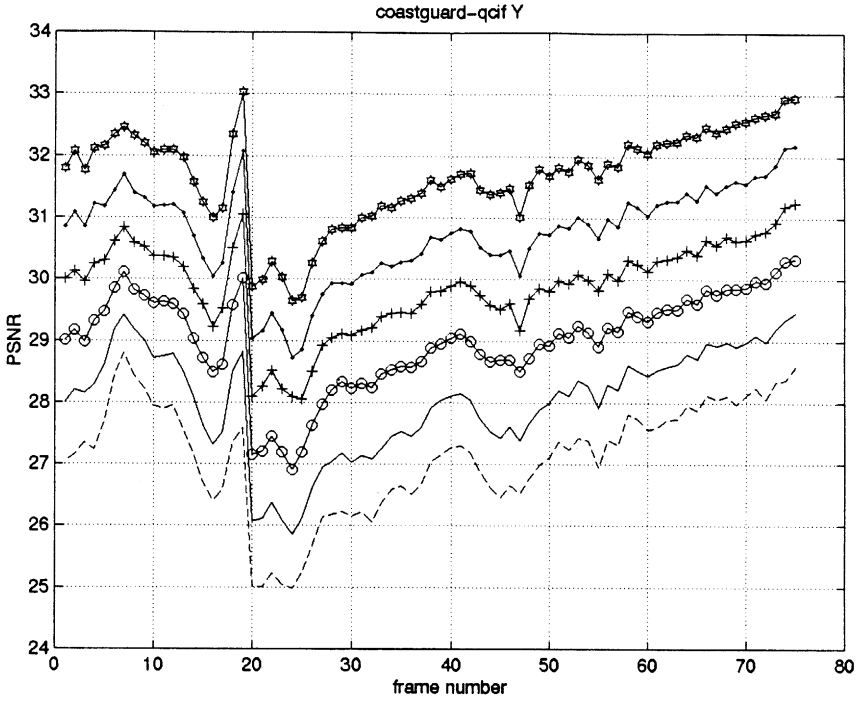

Fig. 11. Plots for the PSNR values of the luminance pictures of the 'coastguard' sequence (see an example in Fig. 10). The lower plot represents the PSNR performance of the base-layer coded at $R = 24$ kbps. The plots with the higher PSNR values are for enhanced sequences decoded using enhancement bit-rates R, $2R$, $3R$, $4R$ and $5R$, in an ascending order. It is important to note that only a single enhancement-layer wavelet stream was generated, and therefore all of the enhancement pictures were decoded from the same stream in a fine-granular way.

activity is very important in that respect. Keeping with the long and successful tradition of MPEG, this activity will ensure that a very robust and efficient FGS coding tool will be supported. The level of interest that this new activity has generated within the MPEG-4 video committee is an important step in that direction.

Another crucial element for the success of an FGS video coding method is the reliable and efficient streaming of the content. In particular, reliable delivery of the base-layer video is of prime importance. In that regard, the deployment of a streaming solution with packet-loss handling mechanism is needed. In the next section, we will focus on developing the re-transmission based packet-loss handling mechanism (mentioned earlier in the document) for the delivery of the base-layer video. We will also illustrate the effectiveness of that approach. Due to the fine-granularity of the scalable video scheme we are using, a packet loss of enhancement layer video only impacts the particular frame experiencing the loss. Consequently, we only provide packet-loss recovery for the base layer. Therefore, for the remaining of the document we will focus on describing a base-layer video buffer model which supports re-transmission of lost packets while preventing underflow[9] events from occurring.

5. Integrated transport-decoder buffer model with re-transmission

5.1. Background

Continuous decoding and presentation of compressed video is one of the key requirements of

[9] Underflow occurs when all pieces of data, which are needed for decoding a picture, are not available at the receiver at the time when the picture is scheduled to be decoded.

real-time multimedia applications. In order to meet this requirement, a decoder-encoder buffer model is normally used to ensure that underflow and overflow events do not occur. These constraints limit the size (bitwise) of pictures that enter the encoder buffer. The constraints are usually expressed in terms of encoder-buffer bounds, which when adhered to by the encoder, guarantee continuous decoding and presentation of the compressed video stream at the receiver.

Fig. 12 shows an ideal encoder-decoder buffer model of a video coding system. Under this model, uncompressed video pictures first enter the compression engine of the encoder at a given picture rate. The compressed pictures exit the compression engine and enter the video encoder buffer at the same picture rate. Similarly, the compressed pictures exit the decoder buffer and enter the decoding engine at the same rate. Therefore, the end-to-end buffering delay (i.e. the total delay encountered in both the encoder and decoder buffers) is constant. However, in general, the same piece of compressed video data (e.g. a particular byte of the video stream) encounters different delays in the

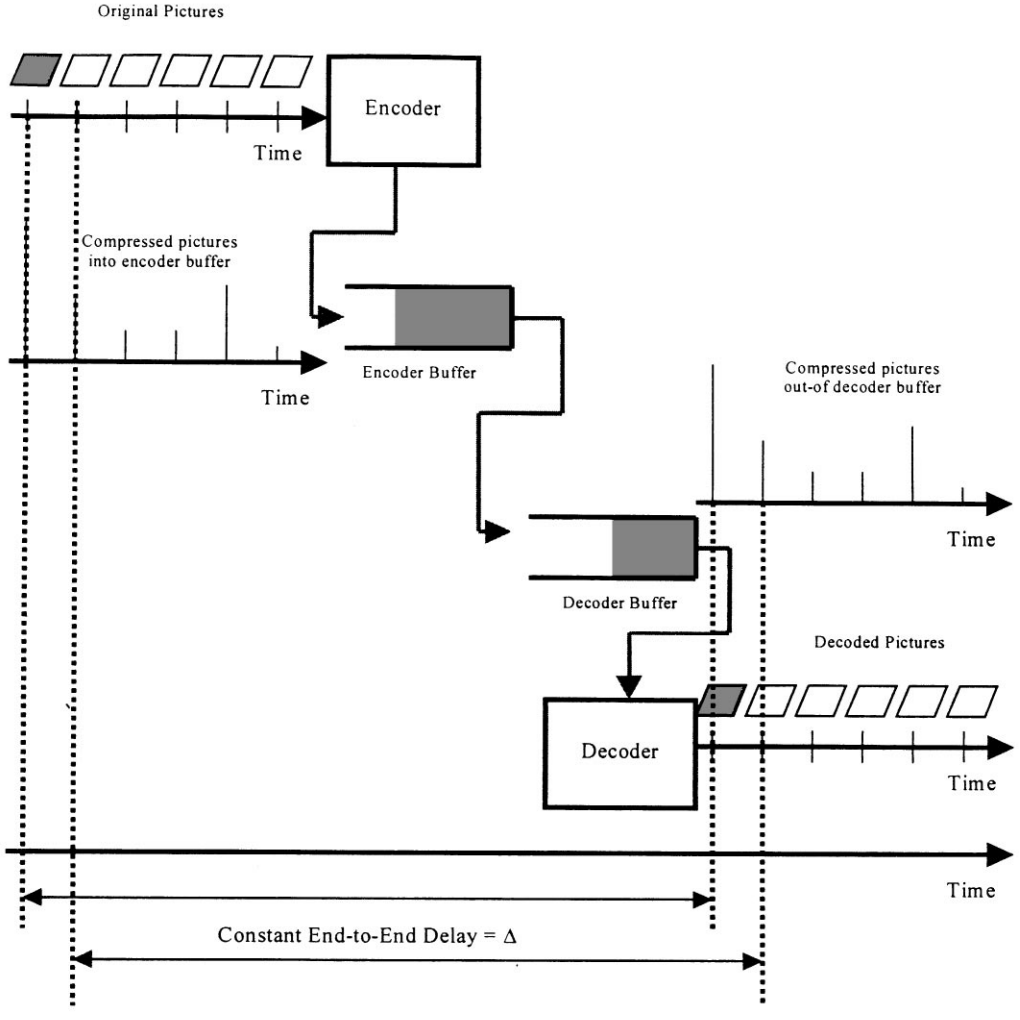

Fig. 12. An ideal, encoder-decoder buffer model of a video coding system.

encoder and decoder buffers. Encoding and decoding take zero time under this model.

The encoder buffer bounds can be expressed using either discrete-time summation [14,21] or continuous-time integration [36]. Here we choose the discrete-time domain analysis. First, let Δ be the end-to-end delay (i.e. including both encoder and decoder buffers, and the channel delay δ_c) in units of time. For a given video coding system, Δ is a constant number that is applicable to all pictures entering the encoder-decoder buffer model. To simplify the discrete-time expressions, it is assumed that the end-to-end buffering delay $\Delta T = \Delta - \delta_c$ is an integer-multiple of the frame duration T. Therefore, $\Delta N = (\Delta - \delta_c)/T$ represents the buffers' delay in terms of the number of video pictures.[10]

Let $r^e(i)$ be the data rate[11] at the output of the encoder during frame-interval i. If $r^d(i)$ is the data rate at the input of the decoder buffer, then based on this ideal model $r^e(iT) = r^d(iT + \delta_c)$. In addition, based on the convention we established above this expression is equivalent to $r^e(i) = r^d(i)$. The encoder buffer bounds can be expressed as in [14,21]:

$$\max\left(\sum_{j=n+1}^{n+\Delta N} r^e(j) - B^d_{\max}, 0\right) \leqslant B^e(n) \leqslant \min\left(\sum_{j=n+1}^{n+\Delta N} r^e(j), B^e_{\max}\right). \quad (3)$$

$B^d_{\max}$ and $B^e_{\max}$ are the maximum decoder and encoder buffer sizes, respectively. By adhering to the bounds expressed in Eq. (3), the encoder guarantees that the decoder buffer will not experience any underflow or overflow events. Throughout the rest of this document, we will refer to this model as the *ideal* buffer model.

Here we also assume that the encoder starts transmitting its data immediately after the first frame enters the encoder buffer.[12] Therefore, the start-up delay dd_f (which is the delay the first piece of data from the first picture spends in the decoder buffer prior to decoding) equals the end-to-end, encoder-decoder buffer delay: $dd_f = \Delta T = T \cdot \Delta N$.

Two problems arise when applying the above ideal buffer model to non-guaranteed Quality of Service (QoS) networks such as the Internet. First, due to variation in the end-to-end delay between the sender and the receiver (i.e. delay jitter), Δ is not constant anymore. Consequently, in general, one cannot find a constant δ_c such that $r^e(iT) = r^d(iT + \delta_c)$ at all times. Second, there is usually a significant packet loss rate. The challenge here is to recover the lost data prior to the time when the corresponding frame must be decoded. Otherwise, an underflow event will occur. Furthermore, if prediction-based compression is used, an underflow due to lost data may not only impact the particular frame under consideration, but many frames after that. Based on the FGS video scheme employed by our solution, a lost packet in the base layer will impact pictures at both the base and enhancement layers. Therefore, for the remainder of this section we will focus on the development of a receiver buffer model that minimizes underflow events, while taking into consideration the two above problems and the ideal encoder–decoder buffer constraints. The model is based on lost packet recovery using re-transmission.

It has been well established in many published works that re-transmission based lost packet recovery is a viable approach for continuous media communication over packet networks [2,10,22,34]. For these applications, it has been popular to employ a negative automatic repeat request (NACK) in conjunction with re-transmission of the lost packet. All of the proposed approaches take into consideration both the round-trip delay and the delay

[10] Throughout the rest of this document, our time measurements will be in units of frame-duration intervals. For example, using the encoder time reference shown in Fig. 12, the nth picture enters the encoder buffer at time index n. The decoder time reference is shifted by the channel delay δ_c. As noted in previous works (e.g. [14]), smaller time-intervals can also be used within the same framework.

[11] Here we use 'data rate' in a generic manner, and therefore it could signify 'bit', 'byte' or even 'packet' rate. More importantly, $r(i)$ represents the total amount of data transmitted during period i.

[12] This assumption is mainly intended for simplifying the description of the ITD buffer model, and therefore there is no loss in generality.

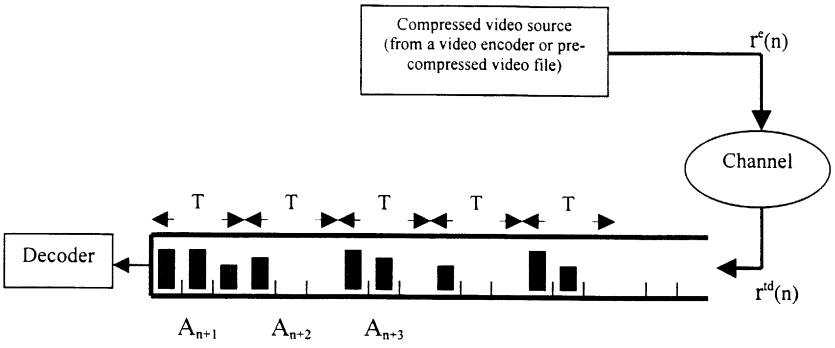

Fig. 13. The basic integrated transport-decoder buffer model.

jitter between the sender and the receiver(s). For example, in [10], an end-to-end model with re-transmission for packet voice transmission is developed. The model is based on the paradigm that the voice data consists of silence and talkspurt segments. The model also assumes that each talkspurt consists of a fixed number of fixed-size packets. Although this model can be applicable for voice data, it is not general enough to capture the characteristics of compressed video (which can have variable number of bytes or packets per video frame).

Here we develop a receiver buffer model that takes into consideration both transport delay parameters (end-to-end delay and delay jitter) and the video encoder buffer constraints described above. We refer to this model as the Integrated Transport Decoder (ITD) buffer model. One key advantage of the ITD model is that it eliminates the separation of a network-transport buffer, which is typically used for removing delay jitter and recovering lost data, from the video decoder buffer. This reduces the end-to-end delay, and optimizes the usage of receiver resources (such as memory).

5.2. The basic ITD buffer model

One of the key questions that the ITD model addresses is: how much video data a receiver buffer must hold at a given time in order to avoid an underflow event at a later time? The occupancy of a video buffer is usually expressed in terms of data units (bits, bytes, etc.) at a given time instance. This however does not match well with transport layer, ARQ based re-transmission methods that are based on temporal units of measurements (e.g. round-trip delay for re-transmission). The ITD integrates both a temporal and data-unit occupancy models. An ITD buffer is divided into temporal segments of T duration each. A good candidate for the parameter T is the frame period of a video sequence. The data units (bits, bytes or packets) associated with a given duration T is buffered in the corresponding temporal segment. This is illustrated in Fig. 13. During time interval n, the nth access unit[13] (A_n) is being decoded, and access unit A_{n+1} is stored at the temporal segment nearest to the buffer output. Therefore, the duration it takes to decode or display an access unit is the same as the duration of the temporal segment T. During the time-interval n, the rate at which data enters the ITD buffer is $r^{td}(n)$.

Each temporal segment holds a maximum number of packets $K_{\max}$. And, each packet has a maximum size of $b_{\max}$ (in bits or bytes). Therefore, if $S_{\max}$ represents the maximum size of an access unit, then $S_{\max} \leqslant K_{\max} b_{\max}$. Here we assume that packetization is done such that each access-unit commences with a new packet.[14] In other words,

[13] Here we use the notion of an access unit which can be an audio frame, a video frame, or even a portion of a video frame such as Group of Blocks (GOB).

[14] The model here is not dependent on the particular packet type (IP, UDP, RTP, ATM cells, etc.). For Internet streaming, RTP packets may be a good candidate. Regardless what packet type one chooses, the packetization overhead must be taken into consideration when computing key parameters such as the data rates, packet inter-arrival times, etc.

the payload of each packet belongs to only one access unit.[15]

There are two measures we use to express the occupancy of the ITD buffer $B^{td}(n)$ at time index n:

$$B^{td}(n) = (B^{a}(n), B^{b}(n)),$$

$B^{a}(n)$ represents the number of *consecutive-and-complete* access units in the buffer at the beginning of interval n. Temporal segments containing partial data are not counted, and all segments following a partial segment are also not counted even if they contain a complete, access-unit worth of data. Hence, $TB^{a}(n)$ represents how much video in temporal units (e.g. seconds) that the ITD buffer holds at time index n (without running into an underflow if no more data arrives). Here we use the following convention for labeling the ITD buffer temporal segments. When access unit A_n is being decoded, the temporal segment holding access unit A_{n+i} is labeled the ith temporal segment. The temporal segment with index $i = 1$ is the nearest to the output of the buffer. Therefore, and assuming there are no missing data, temporal segments $i = 1, 2, \ldots, B^{a}(n)$ are holding complete access units.

$B^{b}(n)$ is the total consecutive (i.e. without missing access units or packets) amount of data in the buffer at interval n. Therefore, if S_n denotes the size of access unit n, then the relationship between B^{a} and B^{b} can be expressed as follows:

$$B^{b}(n) = \sum_{j=n+1}^{n+B^{a}(n)} S_j + U_{B^{a}(n)+1}. \tag{4}$$

$U_{B^{a}(n)+1}$ is the partial (incomplete) data of access unit $A_{n+B^{a}(n)+1}$ which is stored in temporal segment $B^{a}(n)+1$ at time index n.

5.3. The ITD model with re-transmission

Four processes influence the occupancy of the ITD buffer when re-transmission is supported: (a) the process of outputting one temporal segment (T) worth of data from the buffer to the decoder at the beginning of every time-interval n, (b) the detection of packet loss(es) and transmission of associated NACK messages, (c) the continuous arrival of primary packets (i.e. not re-transmitted), and (d) the arrival of the re-transmitted packets.

Moreover, the strategy used for detecting packet losses and transmitting NACK messages can have an impact on the buffer model. For example, a single-packet loss detection and re-transmission request strategy can be adopted. In this case, the system will only attempt to detect the loss events on a packet-by-packet basis, and then send a single NACK for each lost packet detected. Another example arises when a multiple-packet loss detection and re-transmission request strategy is adopted. In this case, the system attempts to detect multiple lost packets (e.g. that belongs to a single access unit), and then send a single re-transmission request for all lost packets detected.

Here we derive ITD buffer constraints that must be adhered to by the receiver to enable any generic re-transmission strategy. Let T_L represents the minimum duration of time needed for detecting a predetermined number of lost packets. In general, T_L is a function of the delay jitter between the sender and the receiver[16] due to data arriving later than expected to the ITD buffer. Let T_R represents the minimum amount of time needed for recovering a lost packet after being declared lost by the receiver. T_R includes the time required for sending a NACK from the receiver to the sender and the time needed for the re-transmitted data to reach the receiver (assuming that the NACK and re-transmitted data are not lost). Therefore, T_R is a function of the round-trip delay between the receiver and the sender.

To support re-transmission of lost packets, video data must experience a minimum delay of $(T_L + T_R)$ in the ITD. Let the minimum delay experienced by any video data under the *ideal*

[15] As discussed later, the extension of the ITD model to the case when each packet contains an integer number of access units is trivial. This later case could be typical for audio packetization.

[16] Other factors that can influence the parameter T_L are: variation in the output data rate due to packetization (i.e. packetization jitter), the inter-departure intervals among packets transmitted from the sender, and the sizes of the packets. The important thing here is that T_L must include any time elements needed for the detection of a lost packet at the receiver.

decoder buffer model be dd_{min}. Therefore, the amount of delay that must be added to the minimum ideal delay in order to enable re-transmission is

$$\Delta_R \geqslant u(T_L + T_R - dd_{min}), \quad (5)$$

where $u(x) = x$ for $x > 0$, and $u(x) = 0$ for $x \leqslant 0$.

The delay Δ_R must be added to all data to ensure the continuous decoding and presentation of video. Therefore, if Δ_{ideal} is the ideal encoder–decoder buffer delay,[17] then the total encoder-ITD buffer model delay is

$$\Delta_{TOT} = \Delta_{ideal} + \Delta_R \geqslant \Delta_{ideal} + u(T_L + T_R - dd_{min}). \quad (6)$$

5.3.1. ITD buffer bounds

Based on the constraints described above, we derive here the ITD lower and upper bounds that must be maintained at all times. Let B^a_{min} be the minimum number of temporal segments that must be occupied with data in the ITD buffer in order to enable re-transmission and prevent an underflow event. Therefore, and in the absence of lost packets and delay jitter, at any time index n, the ITD buffer occupancy must meet the following:

$$TB^a(n) \geqslant TB^a_{min} = T_L + T_R. \quad (7)$$

Let dd_{max} be the maximum decoding delay experienced under the ideal buffer model. Hence, $dd_{max} \leqslant \Delta_{ideal}$. Consequently, and also in the absence of lost packets and delay jitter, the ITD buffer has to meet the following:

$$TB^a(n) \leqslant dd_{max} + u(T_L + T_R - dd_{min}) \leqslant \Delta_{ideal} + u(T_L + T_R - dd_{min}). \quad (8)$$

Therefore, in the absence of lost data and delay jitter, the ITD buffer occupancy is bounded as follows:

$$T_L + T_R \leqslant TB^a(n) \leqslant dd_{max} + u(T_L + T_R - dd_{min}). \quad (9)$$

[17] Δ_{ideal} is the same as ΔT of the previous section. Here, however, we want to clearly distinguish between the delay associated with the ideal case from the delay of the ITD model.

Taking delay jitter into consideration, the buffer occupancy can be expressed as

$$T_R \leqslant TB^a(n) \leqslant dd_{max} + u(T_L + T_R - dd_{min}) + T_E, \quad (10)$$

where T_E is the delay jitter associated with packets arriving earlier than expected to the ITD buffer. Therefore, if B^a_{max} is the maximum number of temporal segments that the ITD buffer can hold, then

$$TB^a_{max} \geqslant dd_{max} + u(T_L + T_R - dd_{min}) + T_E$$

or

$$B^a_{max} \geqslant \left\lceil \frac{dd_{max} + u(T_L + T_R - dd_{min}) + T_E}{T} \right\rceil. \quad (11)$$

5.4. ITD buffer-based re-transmission algorithm

Here we describe a re-transmission algorithm based on the ITD buffer model. To simplify the description of the algorithm, we assume that T_L and T_R are integer-multiples of the duration T. Let $N_R = T_R/T$ and $N_L = T_L/T$. Furthermore, we first describe the algorithm for the scenario when the minimum decoding delay under the ideal model is zero: $dd_{min} = 0$, and the maximum decoding delay is equal to the ideal end-to-end buffering delay: $dd_{max} = \Delta_{ideal}$. In this case, the extra minimum delay that must be added to the ideal buffer is $T_L + T_R$. This corresponds to $N_L + N_R$ of temporal segments. From Eq. (11), the total number of temporal segments needed is

$$B^a_{max} \geqslant N_L + N_R + \lceil (T_E + dd_{max})/T \rceil. \quad (12)$$

Since the maximum decoding delay $dd_{max} (= \Delta_{ideal} = \Delta T)$ corresponds to ΔN temporal segments, then

$$B^a_{max} \geqslant N_R + N_L + \Delta N + N_E, \quad (13)$$

where $N_E = \lceil T_E/T \rceil$.

Based on Eq. (13), one can partition the ITD buffer into separate segments. Fig. 14 shows the different regions of the ITD buffer model under the above assumptions. The two main regions are:

1. the *ideal-buffer region* which corresponds to the buffer area that can be managed in the same way that an ideal video buffer is managed.

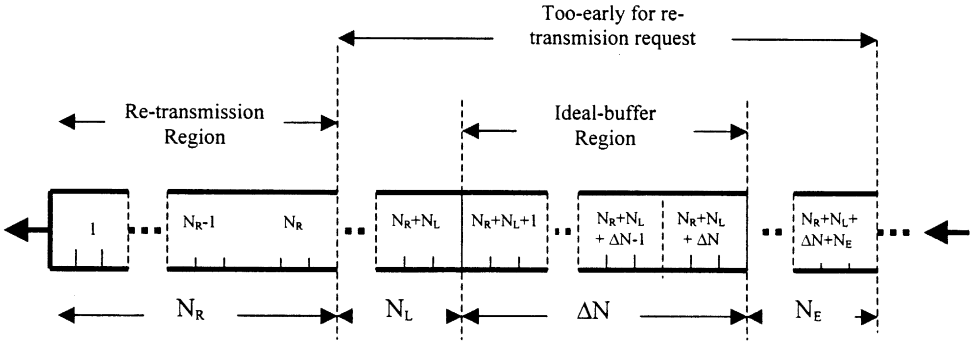

Fig. 14. The different segments of the ITD buffer under the case of a set of extreme values for the ideal delay parameters: $dd_{min} = 0$ and $dd_{max} = \Delta_{ideal}$.

2. the *re-transmission region* that corresponds to the area of the buffer when requests for re-transmission should be initiated, and the re-transmitted data should be received (if they do not encounter further losses).

It is important to note that the two above regions may overlap depending on the values of the different delay parameters (dd_{min}, T_R, T_L). However, for the case $dd_{min} = 0$, the re-transmission and ideal-buffer regions do not overlap. Furthermore, as data move from one temporal segment to another, request for re-transmission must not take place prior to entering the re-transmission region. Therefore, we refer to all temporal segments that are prior to[18] the re-transmission region as the 'too-early-for re-transmission request' region (as shown in Fig. 14).

Before describing the re-transmission algorithm, we define one more buffer parameter. Under the ideal model, the initial decoding delay dd_f represents the amount of delay encountered by the very first piece of data that enters the buffer prior to the decoding of the first picture (or access unit). This delay is based on, among other things, the data-rate used for transmission for the duration dd_f. In the ideal case, this rate also represents the rate at which the data enters the decoder buffer as explained earlier. Let B_0^d be the buffer occupancy of the ideal model just prior to the decoding of the first access unit. Therefore,

$$B_0^d = \sum_{j=1}^{\Delta N} r^e(j). \tag{14}$$

We refer to the data that is associated with Eq. (15) as the 'start-up-delay' data.

The re-transmission algorithm consists of the following procedures:

1. The *ideal-buffer region* is first filled until all data associated with the start-up delay are in the buffer.[19] This condition is satisfied when

$$\sum_{k=N_R+N_L+1}^{N_R+N_L+\Delta N} B_k = B_0^d, \tag{15}$$

   where B_k is the amount of data stored in temporal segment k at any instant of time.
2. After Eq. (15) is satisfied, the content of all temporal segments are advanced by one segment toward the buffer output. Subsequently, this process is repeated every T units of time. Therefore, after $N_L + N_R$ periods of T (i.e. after $T_L + T_R$), the first access unit will start being decoded. This time-period (i.e. at the beginning of which the first access unit is decoded) is

[18] Here 'prior to' in the sense of the first-in-first-out order of the buffer.

[19] This step has to take into consideration that lost events may occur to the 'start-up-delay' data. Therefore, these data may be treated in a special way by using reliable transmission (e.g. using TCP) for them.

labeled T_1. Hence, at the beginning of any time period n, access unit A_{n+k} is moved to temporal segment k.

3. As data move into the *re-transmission region*, any missing data in temporal segment N_R must be considered lost. This condition occurs when

$$B_{N_R}(n) < S_{n+N_R}, \tag{16}$$

where $B_{N_R}(n)$ is the amount of data in temporal segment N_R at time period n, and S_j is the size of access unit j. When missing data are declared lost, then a re-transmission request is sent to the sender.

4. As re-transmitted data arrive at the ITD buffer, they are placed in their corresponding temporal segments. Based on the buffer model, and assuming the re-transmitted data are received, then the re-transmitted data will arrive prior to the decoding time of their corresponding access units.

As explained above, this description of the algorithm was given for the case when $dd_{min} = 0$. For the other extreme case when $dd_{min} \geqslant T_L + T_R$, the *re-transmission region* of the ITD buffer totally overlap with the *ideal-buffer region* as shown in Fig. 15. In this case, the algorithm procedures described above are still valid with one exception. Here, after all of the data associated with the start-up-delay arrives, the first access unit will be decoded immediately without further delays. In the general case when dd_{min} is between the two extreme cases (i.e. when $0 < dd_{min} < T_L + T_R$), there will be an additional delay of $(T_L + T_R - dd_{min})$.

In general, the effectiveness of the re-transmission algorithm in recovering lost packets depends, among other things, the values used for T_L and T_R, and the rate at which the server transmits the data. In the following section, we will address the latter issue and describe a simple mechanism for regulating the streaming of data at the server output. Then, we will address the impact of the delay parameters on the effectiveness of the re-transmission scheme and show some results for real-time streaming trials conducted over the Internet.

5.5. Regulated server rate control

In order to avoid buffer overflow and underflow, it is important that the stream be transmitted at the rate at which it was created. Due to packetization (e.g. RTP), the rate at which the server outputs the data may differ from the ideal desired rate (i.e. packetization jitter). Therefore, it is important to minimize this rate variation. In addition, it is important to stream the data in a regulated manner to minimize network congestion and packet-loss events.

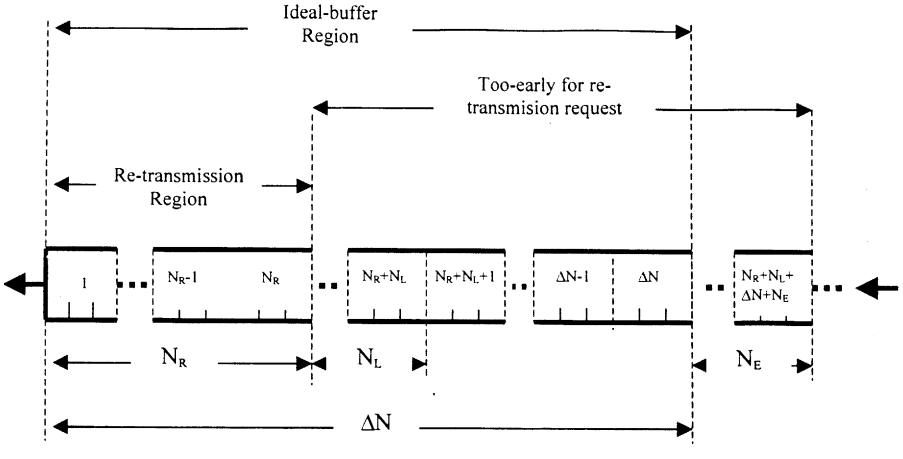

Fig. 15. The different segments of the ITD buffer under the case when ($dd_{min} \geqslant T_R + T_L$) and $dd_{max} = \Delta_{ideal}$. In this case, the re-transmission related delays can be observed by the end-to-end, ideal buffering delay Δ_{ideal}.

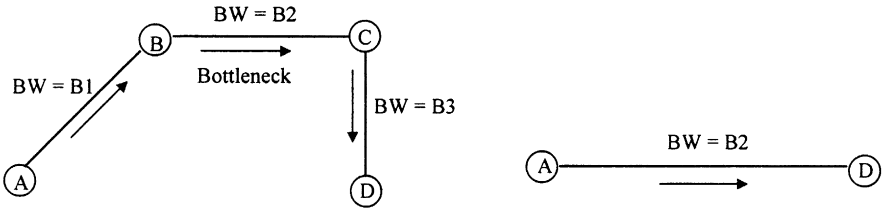

Fig. 16. Equivalent network based on a bottleneck connection with a bandwidth B2.

Owing to the delays associated with Transport Control Protocol (TCP), User Datagram Protocol (UDP) is usually the protocol of choice for streaming real-time media over the Internet. Since UDP does not inherently exercise flow control, improperly designed UDP applications can be a threat to existing applications like ftp, telnet, etc. that run atop socially-minded protocols like TCP. Besides, poorly designed UDP applications can congest the network, and with the proliferation of streaming applications, this could eventually result in major congestion in the Internet.

In our system, we regulate the rate of the streaming UDP application to match that of the bottleneck link between the sender and the receiver. The mechanism by design avoids congestion by injecting a packet into the network only when one has left it. Besides reducing the chance of packet loss due to congestion, this method allows the application to achieve rates that are very close to the encoded rate. If there is a means of communicating information from the receiver to the sender, rates can be changed during the course of the streaming with ease.

We assume that we have a measure of the bottleneck-bandwidth[20] (the maximum rate at which the application can inject data into the network without incurring packet loss), and the round-trip time (RTT). The receiver can get an approximate measure of the bottleneck bandwidth by counting the number of bits it receives from the sender over a given duration. This measure can be communicated back to the sender (e.g. through RTCP). In the event that there is no communication from the receiver to the sender, this method can still be used if the bandwidth does not significantly change during the course of the application. In the case of streaming scalable content, we transmit only the base-layer and portion of the enhancement layer that will satisfy the bottleneck requirements.

The left of Fig. 16 shows three links in the network between the sender A and the receiver D. The bottleneck link is the link between the nodes B and C and the bottleneck bandwidth is B2. B2 is thus the maximum rate at which data will be removed from the network and is hence the rate at or below which the application must transmit the data to avoid packet loss. The figure on the right denotes the equivalent network diagram. For the rest of this document, we assume that we have a base-layer stream that matches or is less than the bottleneck bandwidth, B2. Therefore, if N is the number of temporal units (in frames) over which the bandwidth B2 is measured, we assume the following is true for any K:

$$\left(\sum_{j=K}^{K+N} r^e(j)\right) \Big/ N \leqslant \mathrm{B2}.$$

Let the maximum number of bits read off the network in a time interval T be dictated by the bottleneck bandwidth B and is given by BT. This is also the amount of data that the server can inject into the network in the time interval T. In each time interval T, we inject as many packets into the network so as to come as close as possible to the bit-rate $r^e(i)$ at which the base-layer is coded. Moreover, in practice, the available bandwidth B may change over time.

If B_i represents the bottleneck bandwidth estimate during the ith time interval, then the remaining bit-rate $\mathrm{RB}_i = (B_i T - r^e(i))$ can be used to

[20] We assume that this measure takes into consideration other users of the network as well.

transmit the enhancement layer video and for serving any re-transmission requests. The re-transmission packets have a higher priority than the enhancement layer video. As explained above, due to the fine-granularity of the enhancement layer, any arbitrary number of enhancement bits can be transmitted depending on the available bandwidth. An example of this scenario is shown in Fig. 17.

This approach thus streams the data in a manner that avoids buffer underflow and overflow events. In addition, we avoid bursting the data into the network thereby reducing the chance of loss.

5.6. *Effectiveness of the re-transmission algorithm*

The ITD buffer re-transmission algorithm was tested over a relatively large number of isolated unicast Internet sessions (about 100 trials). The main objective was to evaluate the effectiveness of our re-transmission scheme as a function of the buffering delays we introduce at the ITD buffer. The key parameters in this regard are the values used for T_L and T_R. As explained earlier, T_L is a function of the delay jitter between the server and the client, and T_R is a function of the round-trip delay. In practice, both T_L and T_R are random variables and can vary widely. Therefore, it is virtually impossible to pick a single set of 'reasonable' values that will give 100% guaranteed performance for recovering the lost packets even if we assume that all re-transmitted packets are delivered to the client. Hence, the only option is to select some values that give a desirable level of performance. Before proceeding, we should identify a good criteria for measuring the effectiveness of our re-transmission scheme. In here, the primary concern is the avoidance of underflow events at the base-layer video decoder. Therefore, we associate the effectiveness of the scheme with the percentage of times that we succeed in recovering lost packets prior to their decode time. If at the decode time of a picture one or more of that picture's base-layer packets are not in the buffer, then this represents an underflow event.

Let t_{ds} and t_{us} represent the packet delays between the server-to-client (downstream) and between the client-to-server (upstream) directions, respectively. The time needed to recover a lost packet (i.e. T_R) using a single-attempt re-transmission can be expressed as $T_R = t_{ds} + t_{us} + C_R$, where C_R accounts for processing and other delays at both the sender and receiver. It has been well

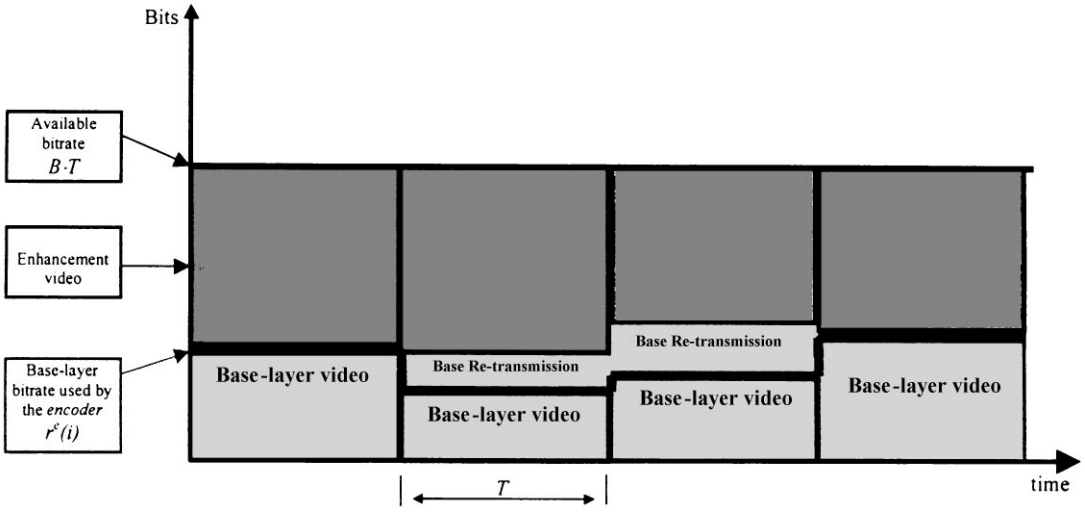

Fig. 17. An example of allocating available bandwidth among the base-layer, enhancement-layer and re-transmitted packets. The base-layer having the highest priority, then re-transmitted data, then enhancement.

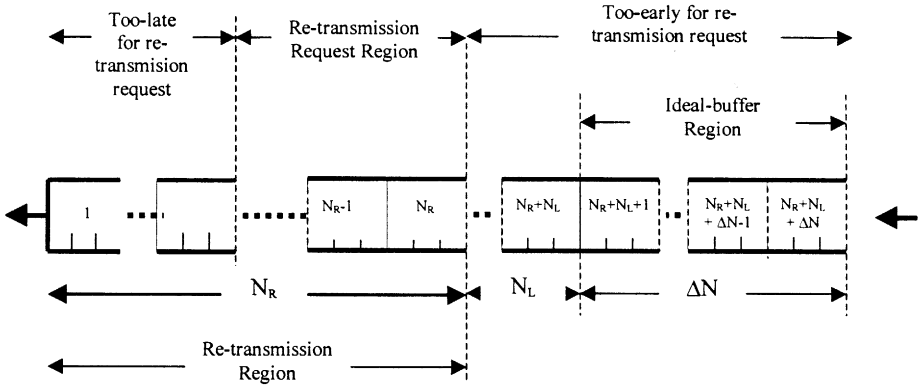

Fig. 18. Partitioning the re-transmission region into a re-transmission request region and a 'too-later for re-transmission request' area of the buffer.

documented that packet delays over the Internet vary in a random manner [33]. Based on the work reported in [29], packet delays over a given Internet session can be modeled using a shifted gamma distribution. However, the parameters needed for characterizing this distribution changes from one path to another, and for a given path changes in time [29,33]. Therefore, and as pointed out in [33], it is difficult to model the delay distribution for a generic Internet path. Here, it suffices to say that the total delay ($T_R + T_L$) introduced by the ITD buffer[21] is a random variable with some distribution function $P_D(t)$. The objective is to choose a minimum value for ($T_R + T_L$) that provides a desired success rate SR for recovering lost packets in a timely manner: $T_R + T_L = D_0$, such that $P_D(D_0) = SR$. Before presenting our results, it is important to identify two phenomena that influence how one would define the success rate of the re-transmission algorithm.

In practice, it is feasible to get into a situation where the buffer occupancy is too low to the extent that a lost packet is detected too late for requesting re-transmission. This is illustrated in Fig. 18 where the re-transmission region now includes a 'too-Late for re-transmission request' (tLfR) area. Of course, this scenario violates the theoretical limits derived in the previous section for the ITD buffer bounds. However, due to changing conditions within the network (e.g. severe variations in the delay or burst packet-loss events), the buffer occupancy may start to deplete within the re-transmission region and toward the tLfR area. In this case, detection of lost packets can be only done somewhere deep within the re-transmission region. If a re-transmission request is initiated within the tLfR area then it is almost certain that the re-transmitted packet would arrive too late relative to its decode time. Therefore, in this case, request for re-transmission is not initiated.

The second phenomenon that influences the success rate of the re-transmission algorithm is the late arrival of re-transmitted packets. In this case, the request for re-transmission was made in anticipation that the packet would arrive prior to the decode time. However, due to excessive delays, the packet arrives after the decode time.

Taking into account the above two observations, we measured our success rate for the re-transmission scheme. We performed the test using a low-bit-rate video coded at around 15 kbps (the MPEG-4 Akiyo sequence) and five frames-per-second. Therefore, the access unit time duration $T = 200$ ms. The sequence was coded with an end-to-end buffering delay of about 2.2 s (i.e. $\Delta N = 11$).

[21] In other words in addition to the ideal buffer delay Δ. Here we are also assuming that the minimum ideal buffer delay dd_{min} is zero.

Therefore, in the absence of packet losses and network jitter, the minimum delay needed for preventing underflow events is 2.2 s. The sequence was looped to generate a 3-min stream for our testing purposes. The three-minute segment was streamed about 100 times using different unicast Internet sessions. The server was located in the New York City (NYC) area and was accessed using a nationwide Internet Service Provider (ISP) with a dial-up modem of 33.6 kbps. The client was also located in the NYC area. In order to test different Internet paths, the test was conducted by having the client dial access numbers located both at the New York area and at the west region of the US. We performed the tests using three different total delays at the receiver (i.e., $\Delta T + T_R + T_L$): 3, 5 and 7 sec. This translates to added delay for ($T_R + T_L$) of 0.8, 2.8 and 4.8 sec, respectively. For each delay scenario, 33 trials were conducted. Therefore, for each delay, a total of about 100 min of video was streamed and tested.

The simulations were performed both in the morning (between 10 am and 11 am EST) and in the afternoon (between 3 pm and 5 pm EST). Please note that these measurements were performed to gauge the performance of the re-transmission mechanism and do not, in any way, characterize the performance of the Internet.

Table 3 shows a summary of the test results. The following is an explanation of the table contents:

1. *Total start-up delay*. This indicates in seconds the delay between when the first packet of the first frame arrives in the buffer and when it is decoded. Therefore, this delay represents the total $\Delta T + T_R + T_L$ as explained above.
2. *Number of re-transmission requests* (*RRs*). This indicates the number of packets that were requested for re-transmission by the receiver.
3. *Number of "too Late for Re-transmission request"* (*tLfR*) *events*. This indicates the number of packets that were also lost but could not be requested for re-transmission because of time constraints (i.e. detected in the tLfR region as explained above).
4. *Total number of lost packets*. This is the total number of lost packets and is the summation of the above two numbers.
5. *Number of re-transmission arrived* (*RA*). This is the total number of re-transmitted packets that were received.
6. *Number of re-transmission arrived too late* (*RAL*). This indicates the number of re-transmitted packets that arrived later than their decode time.
7. *Number of successes*. This is the number of packets that were re-transmitted and received in a timely fashion, and consequently could be used by the decoder.
8. *Success ratio*. This is defined as the ratio of the '*number of successes*' to the '*total number of lost packets*'. Therefore, this measure takes into account both the 'too-late-to-request' and 're-transmission-arrived-too-late' scenarios.

As expected, these results indicate that increasing the start-up delay improves the performance of the algorithm. In addition to improving the overall success ratio, it is clear that the extra delay significantly reduces the tLfR events. Another measure shown in the table is re-transmission request success ratio, which is the ratio of the

Table 3
Summary of the results for testing the success rate of the re-transmission scheme as function of the total delay introduced by the receiver buffer

Total start-up delay (sec)	Number of RRs (NRR)	Number of *LfR* events	Total # of lost packets (NLP)	Number of RA (NRA)	Number of RAL (NRAL)	Number of successes (NS)	Success ratio (SR) (%) NS/NLP	RR success ratio (RSR) (%) NS/NRR
3	566	96	662	544	67	477	72.05	84.27
5	545	9	554	508	31	477	86.10	87.52
7	472	0	472	453	21	432	91.53	91.53

number of re-transmitted packets arrived on time to the number of actual requests were made. Among the two phenomenons explained above, this effectiveness measure takes into consideration only the re-transmitted packets arriving too late. For the 7-second delay case, both success measures gave the same value since there were no tLfR events. It is important to note that the total number of re-transmitted packets arriving at the receiver is less than the total number of re-transmission requests. This is primarily due to losses of the re-transmitted packets themselves over the network. Therefore, one may argue that a more fair measure for the effectiveness of the algorithm is to use the total number of re-transmission packets that actually arrived (NRA) instead of the total number of requested re-transmission (NRR) when computing the RSR measure.

One possible extension of the re-transmission algorithm is to allow multiple re-transmission requests. For example, after a failure in recovering a lost packet in response to the initial request-for-retransmission, a second request is sent to the server (e.g. after waiting for an appropriate amount of time, say T_R). Although a multiple re-transmission-request strategy could improve the overall success rate, the strategy could significantly increase the delay (e.g. roughly doubling the extra delay if two re-transmission requests are used for missing packets). In addition, any extra re-transmission packets will compete (with the normal video traffic) for the available bandwidth. This aspect of the re-transmission scheme could be an interested area for future work.

6. Concluding remarks

In this paper, we described a real-time streaming solution suitable for non-delay-sensitive video applications (e.g. video-on-demand and live TV viewing). We outlined in detail the two main aspects of our proposed streaming solution: an MPEG-4 based scalable video coding method using both a prediction-based base layer and a fine-granular-scalable enhancement layer; and an integrated transport-decoder buffer model with priority re-transmission for the recovery of lost packets, and continuous decoding and presentation of video.

In addition, we gave an overview of a recent activity within MPEG-4 on the development of a fine-granular-scalability coding tool for streaming applications. Results for the performance of our scalable video coding scheme and the re-transmission mechanism were also presented. Actual real-time streaming tests using MPEG-4 video were conducted over the Internet to evaluate the effectiveness of our re-transmission scheme.

The second major aspect of the paper (the buffer model with re-transmission of lost packet recovery scheme) emphasizes the effectiveness of re-transmission based solutions. However, this comes at the expense of relatively high delays. For the type of applications addressed by our solution, these delays, in general, do not present a major issue. Nevertheless, further work is needed to investigate methods for reducing the end-to-end delay in order to enable new applications that may require more responsiveness than the type of delays feasible today with Internet streaming solutions.

Before concluding the paper, we would like to highlight the need for a scalable solution for streaming. One of the main reasons behind the ubiquity of the Internet today is its 'best effort' service model (i.e., no QoS guarantees). This paved the way for the role-out of simple IP-routing-based networks connecting different corners of the world. Meanwhile, there are a great deal of standardization activities and discussions taking place in many committees within and outside the IETF regarding new service models for the Internet such as integrated and differentiated services. Although these service models, when supported, will provide some QoS guarantees, the variability and unpredictability of what bandwidth a user will have when accessing the net will continue to exist. This unpredictability is due to, among other things, variation in consumer access technologies, receiver-platforms, users willingness to pay for premium services (when supported), and the type of QoS guarantees the network can commit to when the connection is maintained. Therefore, we see that

video scalability in general, and FGS in particular can play an important role in supporting multimedia-streaming applications over the Internet today and for some time into the future.

For further reading, see [1,7,11–13,23,26–28,31,42].

Acknowledgements

The authors wish to express their gratitudes to Kiran Challapali, Mihaela van der Schaar, and Mohamed Abdel-Mottaleb from Philips Research, and to Dmitri Loguinov from City University of New York (CUNY) for their valuable feedback on an earlier draft of this manuscript. Special thanks to Percy Yip for his contributions in implementing some of the client software, and to Jim McKinlay for his support in generating the Internet test results. We also would like to express our gratitudes to all of the individuals who participated (and continue to participate) in the MPEG-4 video activity on fine-granular-scalability for their hard work and contributions. Finally, we would like to thank the Guest Editors, Prof. A. Murat Tekalp and Dr. M. Reha Civanlar, and the anonymous reviewers who provided very valuable feedback and many constructive comments on an earlier version of this paper.

References

[1] M. Benetiere, C. Dufour, Matching pursuits residual coding for video fine granular scalability, Contribution to 45th MPEG meeting, Atlantic City, USA, October 1998.

[2] G. Carle, E.W. Biersack, Survey of error recovery techniques for IP-based audio-visual multicast applications, IEEE Network 11 (6) (November/December 1997).

[3] Y. Chen, Summary of mini experiment on fine granular video scalability, Contribution to 45th MPEG Meeting, Atlantic City, USA, October 1998.

[4] Y. Chen, R. Cohen, H. Radha, C. Dufour, W. Li, A. Zakhor, S. Cheung, B. Schuster, J. Liang, Description of Experiment on fine granular video scalability, Contribution to 45th MPEG Meeting, Atlantic City, USA, October 1998.

[5] Y. Chen, C. Dufour, H. Radha, R.A. Cohen, M. Buteau, Request for fine granular video scalability for media streaming applications, Contribution to 44th MPEG Meeting, Dublin, Ireland, July 1998.

[6] Y. Chen, H. Radha, R.A. Cohen, Philips Research USA results of ME on fine granular video scalability, Contribution to 45th MPEG meeting, Atlantic City, USA, October 1998.

[7] S. Cheung, A. Zakhor, Matching pursuit coding for fine granular video scalability, Contribution to 45th MPEG meeting, Atlantic City, USA, October 1998.

[8] W. Davis, J.W. Irza, Cable modems take the early lead, Byte 23 (1) (January 1998).

[9] E.X. Dejesus, How the Internet will replace broadcasting, Byte 21 (2) (February 1996).

[10] B. Dempsey, J. Liebeherr, A. Weaver, On retransmission-based error control for continuous media traffic in packet-switching networks, Comput. Networks ISDN Systems 28 (5) (March 1996) 719–736.

[11] H.R. Deng, M.L. Lin, A type II hybrid ARQ system with adaptive code rates, IEEE Trans. Commun. COM 43 (2/3/4) (February/March/April 1995) 733–737.

[12] S. Floyd et al., A reliable multicast framework for light-weight sessions and application level framing, in: Proceedings of SIGCOMM '95, Vol. 25, ACM, October 1995, pp. 342–356.

[13] M. Handley, An examination of MBONE performance, Tech. Rep., USC, Inf. Sci., Inst. ISI/RR-97-450, January 1997.

[14] C.-Y. Hsu, A. Ortega, A.R. Reibman, Joint selection of source and channel rates for VBR video transmission under ATM policing constraints, IEEE J. Selected Areas Commun. 15 (6) (August 1997).

[15] IETF Internet Draft, Herpel, Balabanian, Basso, Civanlar, Hoffman, Speer, Schulzrinne, RTP payload format for MPEG-4 Elementary Streams, ietf-avt-rtp-mpeg4-00.txt, March 09, 1998.

[16] ISO/IEC 10918-1:1994 | Recommendation ITU-T T.81 (JPEG), Information technology – Digital Compression and coding of continuous-tone still images: Requirements and guidelines, 1994.

[17] ISO/IEC 14496-2, Information technology – coding of audio-visual objects: visual, Committee Draft, ISO/IEC JTC1/SC29/WG11, N2202, March 1998.

[18] ISO/IEC JTC1/SC29/WG11, MPEG-4 version 2 visual working draft rev 3.0, N2212, March 1998.

[19] J. Liang, J. Yu, Y. Wang, M. Srinath, M. Zhou, Fine granularity scalable video coding using combination of MPEG4 video objects and still texture objects, contribution to 45th MPEG meeting, Atlantic City, USA, October 1998.

[20] J.R. Koenen, MPEG-4 multimedia for our time, IEEE Spectrum 36 (2) (February 1999).

[21] T.V. Lakshman, A. Ortega, A.R. Reibman, VBR video: Tradeoffs and potential, Proc. IEEE 86 (5) (May 1998).

[22] X. Li et al., Layered video multicast with retransmission (LVMR): Evaluation of error recovery schemes, in: G. Parulkar (Ed.), NOSSDAV '97, Springer, Berlin, May 1997.

[23] W. Li, Bit-plane coding of DCT coefficients for fine granularity scalability, Contribution to 45th MPEG meeting, Atlantic City, USA, October 1998.
[24] S. McCanne, V. Jackobson, M. Vetterli, Receiver-driven Layered Multi-cast, in: Proceedings of SIGCOMM'96, Stanford, CA, August 1996, pp. 117–130.
[25] S. McCanne, M. Vetterli, V. Jacobson, Low-complexity video coding for receiver-driven layered multicast, IEEE JSAC 16 (6) (August 1997) 983–1001.
[26] E. Miloslavsky, S.-c.S. Cheung, A. Zakhor, SNR scalability using Matching Pursuits, Contribution to 44th MPEG Meeting, Dublin, Ireland, July 1998.
[27] MPEG-4 Requirements Group, MPEG-4 requirements document, Dublin, Ireland, July 1998.
[28] MPEG-4 Requirements Group, MPEG-4 requirements document, Atlantic City, USA, October 1998.
[29] A. Mukherjee, One the dynamics and significance of low frequency components of internet load, Internetworking: Res. Experience 5 (December 1994) 163–205.
[30] J. Nonnenmacher, E.W. Biersack, D. Towsley, Parity-based loss recovery for reliable multicast transmission, in: Proceedings of ACM SIGCOMM'97, Cannes, France, September 1997, pp. 289–300.
[31] J.-R. Ohm, Description of core experiments within MPEG-4 Video, output document from 45th MPEG Meeting, Atlantic City, USA, October 1998.
[32] V. Paxson, End-to-end routing behavior in the internet, in: Proceedings of SIGCOMM'96, August 1996, pp. 25–38.
[33] V. Paxson, End-to-end internet packet dynamics, in: Proceedings of ACM SIG-COM'97 Cong., Cannes, France, September 1997, Vol. 27, No. 4, October 1997, pp. 139–52.
[34] S. Pejhan, M. Schwartz, D. Anastassiou, Error control using retransmission schemes in multicast transport protocols for real-time media, IEEE/ACM Trans. Networking 4 (3) (June 1996) 413–427.
[35] C. Perkins, O. Hudson, Options for repair of streaming media, IETF Network Working Group, RFC: 2354, June 1998.
[36] A.R. Reibman, B.G. Haskell, Constraints on variable bit-rate video for ATM networks, IEEE Trans. Circuits System Video Technol. 2 (December 1992) 361–372.
[37] Said, Pearlman, Image compression using the spatial-orientation tree, IEEE International Symposium On Circuits and Systems, Chicago, IL, May 1993, pp. 279–282.
[38] Said, Pearlman, A new fast and efficient image codec based on set partitioning in hierarchical trees, IEEE Trans. Circuits Systems Video Technol. 6 (June 1996).
[39] Schulzrinne, Casner, Frederick, Jacobson, RTP: a transport protocol for real time applications RFC 1889, Internet Engineering Force, January 1996.
[40] B. Schuster, Fine granular scalability with wavelets coding, contribution to 45th MPEG meeting, Atlantic City, USA, October 1998.
[41] J.M. Shapiro, Embedded image coding using zerotrees of wavelets coefficients, IEEE Trans. Signal Process. 41 (December 1993) 3445–3462.
[42] J.-S. Shin, S-H. Son, D.-S. Cho, Y.-S. Seo, Preliminary results on fine granular video scalability for arbitrary shaped object, contribution to 45th MPEG meeting, Atlantic City, USA, October 1998.
[43] T. Sikora, The MPEG-4 video standard verification model, IEEE Trans. CSVT 7 (1) (February 1997).
[44] T. Sikora, L. Chiariglione, The MPEG-4 video standard and its potential for future multimedia applications, in: Proceeding IEEE ISCAS Conference, Hong Kong, June 1997.
[45] D. Taubman, A. Zakhor, Multirate 3-D subband coding of video, IEEE Trans. Image Process. 3 (September 1994).
[46] J.Y. Tham, S. Ranganath, A. Kassim, Highly scalable wavelet-based video codec for very low bit-rate environment, IEEE Journal on Selected Areas in Commun. 16 (1) (January 1998).
[47] M.Yajnik, J. Kurose, D. Towsley, Packet loss correlation in the Mbone multicast network, in: Proceedings of IEE GLOBECOM, London, UK, November 1996.