Scalable audio coding for compression and loss resilient streaming

M. Sandler and D. Black

Abstract: Current popular Internet audio streaming solutions impose a division between source coding (provided, for example, by MPEG layer III) and channel coding, typically accomplished in the server by means of packet re-transmission. A novel joint source and channel coder that provides packet-loss recovery and continuous bitrate scalability is presented. These functionalities are well suited to streaming audio over third and future generation wireless broadband networks.

1 Introduction

Finally, after a 3-year wait in the UK, 3G is rolling out and the operators are looking to provide high-value data services to boost income in the face of flat revenues from voice services [1]. To compete with wireline in providing the user with a high quality of experience, those services will have to adapt to prevailing channel conditions – available bandwidth and the congestion-induced loss. The technology described in this paper addresses these issues for audio streaming by providing a fully embedded, compressed representation called EZK (which was commercialised by Insonify (www.insonify.com) as ZeroChain), which is fine-grain scalable. This scalability enables enhanced functionality for the evolving Internet by providing novel transport and content delivery solutions.

1.1 Audio streaming

Audio transmission over Internet is usually achieved using the traditional separation of the task into source coding and channel coding. The purpose of source coding is to compress the audio stream as much as possible while affecting the perceived quality of reproduction as little as possible. This process aims to remove both redundant and irrelevant features of the original signal. For this stage, we might use an audio codec, such as MPEG-1 layer 3 (MP3) [2] or MPEG-2 advanced audio codec (AAC) [3].

The purpose of channel coding is to re-insert redundancy into the source-coded audio stream in a way matched to the characteristics of the channel, so as to enable the channel decoder to correct for errors in transmission. This is typically done for packet-switched networks using forward error correction or packet re-transmission. Rarely does a source coder also perform channel coding. The result is that current solutions have severe performance limitations. This shows up, especially in the harsh conditions that will prevail when roaming in 3G networks. The codec described in this paper is an exception, providing source coding and repair and recovery from packet loss. Packet loss is the dominant interference effect over Internet channels.

Another desirable codec feature is bitrate scalability. This allows receivers with widely differing and time-varying connection bandwidths to receive the same audio stream. This is commonly achieved by encoding the signal at several bitrates, storing all the encoded versions on the server, then supplying the appropriate version according to the capacity of each receiver. The alternative uses an embedded codec, in which a single bitstream can be transmitted or decoded at any of a variety of bitrates. The EZK codec described here falls into this category.

Wang *et al.* [4] present a robust scalable framework for existing audio codecs, basing their example on MP3. They use a combination of multi-stage interleaving, layered unequal-sized packetisation and compressed domain error concealment in the receiver. However, the resulting format is not fine-grained in its scalability, offering discrete base layers and enhancement layers in a similar fashion to MPEG-4 coarse-grain scalability [5] (Section 3.1.2).

Zang *et al.* [6] examine a strategy with some similarities to that described in this paper. However, the source-coding strategy is neither as simple as EZK nor as inter-operable as in Wang *et al.* [4]. The audio signal is filtered into four sub-bands, each of which is then modified discrete cosine transform (MDCT)-encoded and subsequently bitplane encoded using a significance refinement process as outlined below. The use of four sub-bands gives precise control over bandwidth and sampling-rate scalability. Also included in this work is biterror protection for wireless channels, as well as a packet-loss recovery scheme.

The work of Li [7] has several similarities with our own. It uses the modified lapped transform, a variant of the MDCT, and bitplane encoding for fine granularity. Compared with the encoder described herein, the main differences are in the provision of an implicit psycho acoustic model which is not sent to the client as side information. Note that the author has not examined issues related to streaming or packet loss.

In this paper, we present an embedded source-coding algorithm using a bitplane encoding and quantisation technique, which underpins new solutions for both packet-loss recovery and scalability.

IEE Proceedings online no. 20050054

doi:10.1049/ip-vis:20050054

Paper first received 25th February and in revised form 20th July 2005

The authors are with the Centre for Digital Music, Department of Electronic Engineering, Queen Mary University of London, Mile End Road, London E1 4NS, UK

E-mail: mark.sandler@elec.qmul.ac.uk

2 Fully embedded, scalable audio coding

The audio codec we describe here (see Leslie and Sandler [8] for more details) is based on embedded zerotree (EZ) quantisation, introduced by Shapiro [9] as EZ wavelet (EZW) coding for image compression. This was shown to provide excellent compression with low complexity. In addition, the process produces a fully embedded code. Said and Pearlman [10] subsequently improved on EZW with set partitioning in hierarchical trees (SPIHT). EZW has been applied to audio compression with good results [11–13].

Fully embedded means that the bits in the bitstream are produced in order of importance and that the coding and, more importantly, the decoding processes may be terminated at any point, yielding a reconstruction quality that is roughly proportional to the length of decoded bitstream. Thus, an embedded code at a certain rate includes all lower-rate codes. This property is highly significant for our purposes, (Section 3). The EZK audio codec uses a frequency-domain decomposition that provides channels of uniform width, typically an MDCT, and is followed by a bit allocation stage consisting of adapted EZW quantisation and a specially designed form of entropy coding. Fig. 1 provides a top-level view of the encoder structure. For fast encoding, the bit allocation may be performed without the psychoacoustic model.

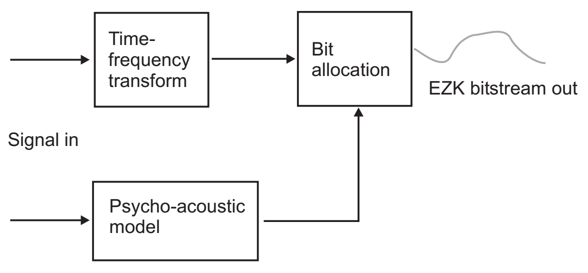

Fig. 1 *EZK encoder structure*

2.1 EZ quantisation for audio

EZ algorithms such as EZW and SPIHT were originally developed for two-dimensional signals (image coding) using a wavelet transform. To use the algorithms in audio coders with wavelet packet (WP) or MDCTs, the main differences to consider are as follows.

2.1.1 Uniform against non-uniform decomposition: The wavelet transform uses non-uniform sub-band decomposition where the low-pass results of previous half-band filtering operations are further decomposed, but the high-pass coefficients are not. In contrast, the WP approach involves further decomposition of both low- and high-pass filter coefficients at each stage, yielding an overall uniform decomposition. The MDCT also provides a uniform decomposition by a direct transform-domain representation for each block of samples within the audio frame [14]. The differences are illustrated in Fig. 2.

2.1.2 Two dimensions against one: Images are two-dimensional and require a two-dimensional transform to yield a matrix of transform coefficients in which both dimensions reflect space and scale. This is illustrated in Fig. 3, where the highest-frequency sub-bands are at the bottom right and the lowest-frequency sub-bands at the top left.

All cells denote transform coefficients that represent the same spatial area in the original image, with the arrows indicating 'heredity' – the low-frequency sub-bands are the 'parents' of the high-frequency sub-bands. In contrast, an audio signal is one-dimensional (in time), and transformation yields a one-dimensional vector of transform coefficients reflecting scale/frequency. However, it is possible to arrange the frequency-domain coefficients from block transforms across several time slots in a two-dimensional matrix (Fig. 4). Here, heredity applies in the frequency domain alone, where all coefficients in the table except the top row (lowest frequency) are 'children' of those at lower frequencies.

All these algorithms follow a pattern of successive refinement, which proceeds in two phases as follows.

1. Significance: it compares the coefficient matrix with a threshold level and produces a sequence of bits or symbols which identifies which coefficients are significant with respect to (i.e. greater than) the threshold. The threshold is reduced (typically halved) following each pass.
2. Refinement: it produces a series of bits that improves the accuracy of representation of coefficients that have already been found to be significant.

Such algorithms were shown in Yu *et al.* [15] to be optimal for sources with Laplacian distribution, which is approximately the case for MDCT and Fourier domain transformations of audio signals.

EZW and SPIHT algorithms were adapted for a coefficient matrix derived from uniform transforms and are now described using pseudo-code.

2.2 EZW algorithm

EZW codes significance of the matrix of coefficients with respect to a threshold that is halved after each pass. This process produces a stream of symbols from a four-symbol alphabet. The four symbols are as follows

POS: the coefficient is significant with respect to the threshold, and positive.
NEG: the coefficient is significant with respect to the threshold, and negative.
IZ (isolated zero): the coefficient is not significant, but has descendants which are.
ZTR (zero tree): the coefficient is not significant and neither are any of its descendants. Therefore there is no need to code a symbol for any of these descendants.

The pseudo-code for EZW is presented in Fig. 5.

2.3 SPIHT algorithm

The SPIHT algorithm [10] codes significance with a stream of bits rather than symbols, using a number of lists of pointers to coefficients in the matrix. These are as follows.

List of significant pixels (LSP), containing pointers to coefficients that have been found to be significant. This list is initially empty.
List of insignificant pixels (LIP), containing pointers to coefficients that are not significant but which have significant descendants. This list initially contains pointers to the top row of the matrix (the parents of all other coefficients).
List of insignificant sets (LIS), containing pointers to coefficients, the significance of whose descendants we

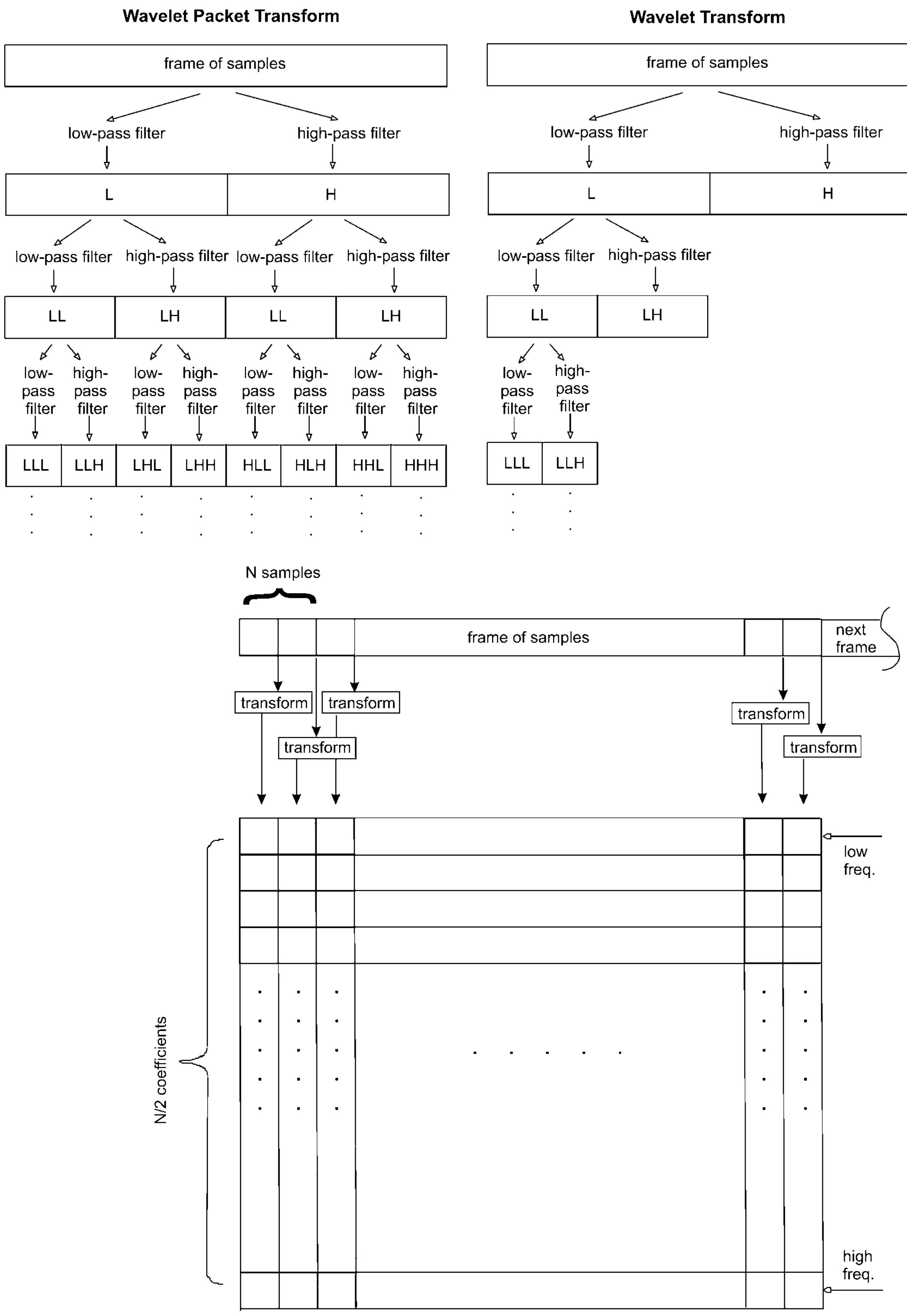

Fig. 2 *Time–frequency decomposition of wavelet transform, WPs and MDCT*

wish to test. Its members are flagged as being of type A if we wish to test the significance of its children and lower generations and of type B if we wish to test the significance of its grandchildren and lower generations. This list initially contains pointers to the top row of the matrix, and these are all flagged as being of type A.

The algorithm is presented in Fig. 6. The main differences between EZW and SPIHT are as follows.

• For EZW, the four symbols produced by the significance stage are POS, NEG, ZTR and IZ. The refinement stage then produces a sequence of bits. SPIHT produces a sequence of bits both to convey significance and for the refinement stage.
• In EZW, for each new significance step, scanning for significant coefficients starts again at the top of the matrix (lowest frequency). This is because the coefficients that were previously insignificant can be tested against the new threshold. Conversely, the SPIHT algorithm avoids returning to the top of the matrix by use of the LIP, each new significance phase beginning with a scan of the LIP. Each scan of the coefficient matrix can then commence from the position where the scan at the previous threshold finished.

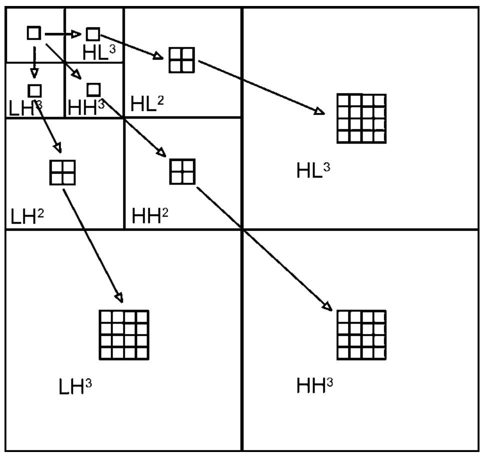

Fig. 3 *Wavelet transform of an image showing heredities through 'sub-bands'*

2.4 EZK algorithm

The motivation for the development of the EZK algorithm was to improve on the compression performance of SPIHT with uniform transform coders (WP and MDCT). Notably, the stream of bits produced by SPIHT when coding a significant coefficient with insignificant parents was unnecessarily long.

We may illustrate this by example in Fig. 4. Here, time progresses leftward, and frequency increases downwards: heredity corresponds to increasing frequency. The timeslot indicated by the arrow shows three 'generations' of insignificant coefficients (denoted '–'), followed by a significant coefficient (denoted '*'). The first coefficient appears in the LIP and also in the LIS as type A. For the SPIHT algorithm, initially a 0 is transmitted because row 1 is insignificant in the scanning of the LIP. Then the LIS is scanned, and 1 is transmitted to show that the coefficient in row 1 has significant descendants. Zero is then transmitted to show that the coefficient in row 2 is not significant and it is added to the LIP. The coefficient in row 1 is then moved to the end of the LIS as type B. When this new entry in the LIS is scanned, a 1 is transmitted because the coefficient in row 1 has significant grandchildren, and the coefficient in row 2 is added to the LIS as type A. This process continues, with coefficients being added to the LIS as type A and type B alternately, until finally the coefficient in row 3 is tested as type A, its child is found to be significant and a 1 is transmitted to show significance, followed by the sign bit. The code for this process is then

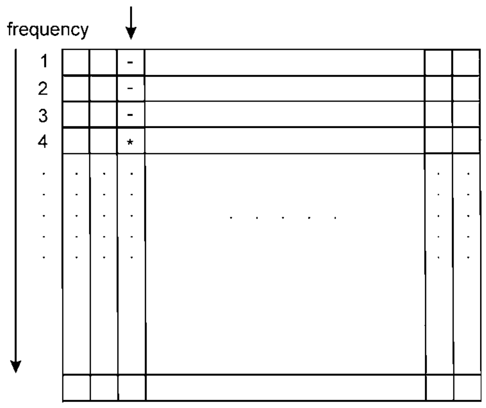

Fig. 4 *Coding insignificance: an example (see text for details)*

```
Max = maximum value in coefficient array
n = ⌊log2(max)⌋
while n>=0
    thresh=2^n
    Significance Stage (Dominant Pass)
    For each time slot
        If timeslot does not yet contain a zerotree symbol
            For each frequency band, from low to high
                If coefficient magnitude for current band and timeslot > thresh
                    Transmit POS or NEG according to sign of coefficient
                    Put value in subordinate list
                    Set matrix value to zero
                Otherwise, if any descendants are > thresh
                    Transmit IZ
                Otherwise,
                    Transmit ZTR
                End
            End
        End
    End

    Refinement Stage (Subordinate Pass)
    For all values in subordinate list,
        Transmit (n–1)th bit of value (where 0th bit is LSB)
    End

    n = n–1
End.
```

Fig. 5 *EZW algorithm*

'010110111X' (where X is the sign bit for the significant coefficient in row 4), a total of ten bits.

The complexity of SPIHT may be necessary for WT coding of images, where use of non-uniform transforms leads to 'family trees' where each coefficient can have more than one descendant. However, for our purposes using uniform transforms, a simplified approach may be used. We may apply the tests on the coefficient in row 1 to determine whether it and its descendants are significant, as before. However, the location of a significant coefficient can be more simply conveyed with a run of 0s followed by a 1. Using this approach, the code for the example in Fig. 4 becomes '01001X', only six bits long. This may be broken down as shown in Table 1.

The lists employed by EZK are as follows.

- *Sig_coeffs*. Analogous to the LSP in SPIHT, it contains pointers to coefficients that have been found to be significant and is initially empty.
- *Insig_coeffs*. Analogous to the LIP in SPIHT, it contains pointers to coefficients that are not significant, but which have significant descendants. It is initially filled with pointers to the top row of the matrix (the parents of all other coefficients).
- *Ts_ptr*. Analogous to the LIS in SPIHT, it contains a pointer for each timeslot (column) in the coefficient matrix. Each one points to the coefficient in that column among whose descendants we shall next be checking for significance. In general, these will point to progressively lower rows (higher-frequency coefficients) as the threshold decreases. We define the remainder of a column as the elements in that column from the member pointed to by *Ts_ptr* downwards.

```
Max=maximum value in coefficient array
n = ⌊log2(max)⌋
while n>=0
    thresh=2^n
    For all members of LIP,
        If coefficient magnitude is significant (> thresh)
            Transmit 1
            Transmit sign bit
            Move pointer to end of LSP
        Otherwise
            Transmit 0
        End
    End

    Significance Stage
    For all members of LIS,
        If type A,
            If coefficient has any significant descendants
                Transmit 1
                If immediate descendant (child) is significant
                    Transmit 1
                    Transmit sign bit
                    Add child to LSP
                Otherwise
                    Transmit 0
                    Add child to LIP
                End
                If coefficient has no grandchildren
                    Remove it from LIS
                Otherwise (significant coefficient must be grandchild or lower)
                    Move coefficient to end of LIS and change to type B
                End
            Otherwise
                Transmit 0
            End
        If type B,
            If coefficient has any significant grandchildren,
                Transmit 1
                Remove coefficient from LIS and add child to end of LIS as type A
            Otherwise
                Transmit 0
            End
        End
    End

    Refinement Stage
    For all coefficients in LSP except those included in last significance pass
        Transmit nth bit of value (where 0th bit is LSB)
    End

    n = n-1
End.
```

Fig. 6 *SPIHT algorithm*

```
Max = maximum value in coefficient array
n = ⌊log2(max)⌋
while n>=0
    thresh=2^n
    For all members of Insig_coeffs,
        If coefficient magnitude is significant (> thresh)
            Transmit 1
            Transmit sign bit
            Move pointer to end of Sig_coeffs
        Otherwise
            Transmit 0
        End
    End

    Significance Stage
    For each time slot (column) of coefficient matrix,
        While remainder of column has any significant members,
            Transmit 1
            For each member of column remainder
                If member is significant
                    Transmit 1
                    Transmit sign bit
                    Add member to Sig_coeffs
                    Set Ts_ptr(column) to point to row below member
                Otherwise
                    Transmit 0
                    Transfer to Insig_coeffs
                End
            End
        End
        Transmit 0   (no more significant coefficients in this column)
    End

    Refinement Stage
    For all coefficients in Sig_coeffs,
         Transmit nth bit of value
    OR   Transmit (n-1)th bit of value
    (Allows EZW-style or SPIHT-style refinement)
    End

    n = n-1
End.
```

Fig. 7 *EZK algorithm*

The pseudo-code for the entire EZK algorithm is presented in Fig. 7.

2.4.1 Compression performance: All three algorithms have been implemented and their compression performance compared on a range of real audio segments. For EZW, we have used a two-bit code for the four symbols POS, NEG, ZTR and IZ, with no entropy coding. Results showing that EZK gives the best compression performance of the algorithms can be found in Leslie *et al.* [16].

2.5 EZK with psychoacoustic weighting

The EZK quantisation algorithm has been incorporated into a simple audio codec, as shown in Fig. 1. This employs an MDCT or WP transform and EZK quantisation of the resulting coefficient matrix. The algorithm allows transform

Table 1: Coding of insignificance using EZK

Bit	Explanation
0	coefficient in row 1 is not significant
1	coefficient in row 1 has significant descendants
0	coefficient in row 2 is not significant
0	coefficient in row 3 is not significant
1	coefficient in row 4 is significant
X	sign bit for coefficient in row 4

lengths from 64 to 1024 points within an overall 1024-sample frame length, thus yielding coefficient matrices with different aspect ratios. We found in Leslie *et al.* [16] that coding performance using an MDCT with a fixed transform length is generally optimised with a 256-point transform, corresponding to a coefficient matrix with 128 rows and eight columns.

The performance of this codec was found in Leslie *et al.* [16] to be superior to MPEG-audio layer I and comparable to layer II, despite the much lower complexity of EZK. Both MPEG codecs employ psychoacoustic modelling to perform bit allocation such that quantisation noise is least perceptible. Conversely, the raw EZK algorithm effectively allocates bits to the highest-magnitude coefficients, the effect of which is to distribute quantisation noise uniformly among frequency bands (i.e. the quantisation noise tends to a white spectrum). Such a noise distribution may be sub-optimal from a psychoacoustic perspective.

It is clearly not possible to perform bit allocation in the manner of MPEG coders. However, it is possible to use a psychoacoustic model to guide a weighting of the frequency bands. The encoder block diagram using this idea is given in Fig. 8. This scheme ensures that the coefficients presented to the EZK algorithm are reduced in relative magnitude where the masking effect is greatest and increased in relative magnitude where masking is least. When the coefficients are subject to an inverse weighting at the decoder, the overall quantisation noise characteristic will then follow the masking threshold rather than being uniform with frequency.

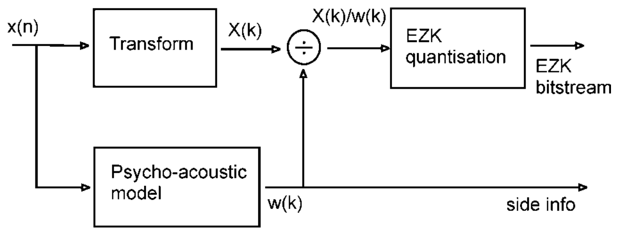

Fig. 8 *EZK encoder with psychoacoustic weighting*

The psychoacoustic model employed in our experimental coder is broadly guided by the principles outlined in Brandenburg *et al.* [2].

1. an FFT of the audio frame is converted into a power spectrum
2. a critical band analysis is performed to determine the energy in each critical band
3. a spreading function is applied in order to take account of masking across critical bands
4. the result is compared to the absolute threshold function across all frequencies, the maximum of which is output as the overall masked threshold function.

Psychoacoustic weighting was implemented as described and tested using a number of signals across a range of bitrates.

The results for a number of MDCT lengths at a bitrate of 64 kbps are shown in Table 2, where a '√' indicates that the reconstruction with psychoacoustic weighting sounds better than without, 'X' that it sounds worse, and '–' that it sounds about the same. The test signals are taken from the MPEG SQAM set (http://sound.media.mit.edu/mpeg4/audio/sqam/).

The results show that psychoacoustic weighting works well for the 256-point transform, although results at other transform lengths are not as good. We assume that the psychoacoustic weighting process is somehow acting to make EZK less efficient. If the weighting has the effect of increasing the magnitude of high-frequency coefficients relative to lower-frequency coefficients, then it will tend to increase the average length of runs of zeros when coding insignificance in the coefficient matrix.

This hypothesis was tested by measuring the proportion of the total coded file size taken up by runs of zeros used to code insignificance. The results are presented in Table 3, where the first column repeats the results from Table 2. The 'No PA' columns correspond to percentages of the coded files, which are taken up by zero runs for coders without psychoacoustic weighting, whereas columns labelled 'With' correspond to percentages for coders with psychoacoustic weighting. The results indicate that the amount of the file dedicated to runs of zeros is greater on average with psychoacoustic weighting, thereby impairing the compression performance of the EZK algorithm.

Table 2: Results for EZK codec with psychoacoustic weighting

Signal	64-point	256-point	512-point
Castanets	X	√	√
Voice	X	√	√
Harpsichord	X	√	–
Pitch pipe	X	X	X

Table 3: Percentages of coded files taken up by zero runs

Signal	Better	No PA	With
64-point			
Castanets	X	12.1	12.5
Voice	X	14.9	16.8
Harp	X	10.6	15.5
Pitch pipe	X	5.8	8.1
256-point			
Castanets	√	29.5	29.1
Voice	√	36.1	37.0
Harp	√	21.8	27.2
Pitch pipe	X	28.9	27.3
512-point			
Castanets	√	29.7	30.3
Voice	√	39.1	39.4
Harp	–	23.0	30.7
Pitch pipe	X	29.7	34.5

We have also compared the EZK codec with the Fraunhofer MP3 codec at three bitrates. Typical results of segmental signal-to-noise ratio (SSNR) are shown in Table 4. We see that the EZK codec performs better at all bitrates.

In informal listening tests, the EZK codec also compares well with MPEG AAC, especially at low bitrates. Note that EZK has the advantages of speed (computational cost) and flexibility over both MP3 and AAC.

3 Bitrate scalable streaming and packet-loss recovery

The fully embedded property of EZK bitstreams means that if, for instance, we encode audio at 64 kbps, we may take the first half of each encoded frame, and we will automatically have a version of the signal exactly as if it had been encoded at 32 kbps. That is, transcoding is achieved simply by truncating a high-rate representation.

This is a significant difference from most audio codecs, which produce structured bitstreams, made up of scale-factors, quantisation, coefficients and so on. If, for example, we take the first half of each MP3 frame, it is impossible to decode anything. The embedded property of EZK underpins the two key features fine-grain bitrate scalable streaming and packet-loss recovery.

3.1 Fine-grain bitrate scalable streaming

The essence of bitrate scalability for streaming audio is the ability to alter the bitrate delivered by a connection in response to network conditions. In the preferred mode of

Table 4: Performance comparison between EZK and MP3 codecs

Coder	SSNR		
	32 kbps	64 kbps	128 kbps
EZK	17.71 dB	23.03 dB	30.88 dB
MP3	13.88 dB	17.99 dB	26.06 dB

operation, packets are transmitted from the server to each client and arrive after a short time delay with very few lost. As the network load increases, buffers at the routers between server and client fill up and the delay increases. As the traffic increases still further, the buffers become full and newly arrived packets are dropped.

This is a time-varying change to the capacity of the channel between the server and the client. As the delay increases and packets start to be lost, the capacity decreases. In this situation, it is better to adapt the bitrate of the audio stream to the capacity of the channel than to attempt to continue sending at the higher bitrate and experience loss. Because the load on the network may vary considerably over very short time-scales, the ability to adapt rapidly, and appropriately, is important.

Another operational requirement is that we might need to be able to supply the same compressed signal at different bitrates to suit users on different connections. Other technical solutions meet these requirements in two different ways.

3.1.1 Storing multiple versions: The same material is encoded at several different bitrates and all are stored on the server. One version is served according to users' connection bandwidth and network conditions. This leads to high server-storage requirements and means that there is a noticeable step-change in quality when the bitrate changes. In addition, if the channel bandwidth falls below what is required for the lowest-bitrate version, then playout stops altogether for a buffering operation to be carried out. This is extremely annoying to users.

3.1.2 Coarse-grain embedding/scalability: The MPEG-4 codec provides both small step scalability, such as bit sliced arithmetic coding [17], and large step/coarse-grain scalability. Coarse-grain scalability works by embedding different versions of the material in one file as portrayed in Fig. 9. A low-quality encoding forms a basic layer. This is locally decoded and subtracted from the original signal: the error is then coded. The local decode followed by error encoding is then iterated for as many enhancement layers as desired [5].

This approach only allows for as many different transmittable bitrates as there are layers. Typically, the basic layer might be at 32 kbps, the first enhancement provides another 64 kbps and the last layer another 96 kbps. Not only are the encoding and decoding processes much more complex than EZK bitstreams, but also they are less flexible. Also, the course-grain embedding provides poorer performance than a single (non-embedded) encoding at the same bitrate.

3.1.3 EZK bitrate scalability: This suffers from none of the drawbacks outlined above. Only one file is stored on the server, and the served bitrate may be adjusted as required to precisely suit individual users' connection bandwidths and network conditions. We encode initially for a high bitrate and then transmit a subset of the coded frame according to the current, demanded bitrate. Because of the fully embedded nature of our solution, the decision on delivered bitrate takes place after encoding and is fully flexible. This makes it particularly suitable for transmission control protocol (TCP)-friendly protocols [18].

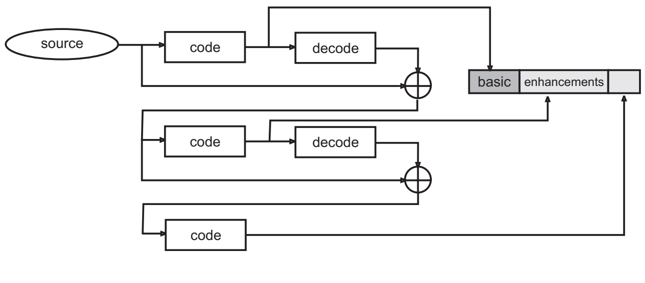

Fig. 9 *Scalable coding in MPEG-4*

3.2 Packet-loss recovery with 'Internet Modulation'

The Internet is a best effort network, which means that there is no guarantee that packets will arrive at their destination. Internet audio streaming requires some method of recovering from such lost packets, if there should not be annoying gaps in the rendered signal. Current solutions include the following.

- Re-transmitting lost packets. This causes a delay, which may be unacceptable for some applications (e.g. videoconferencing), and increases network traffic, which may in turn increase packet losses.
- Error-correction codes. These increase the network traffic, and increase computation, and are only suitable for isolated losses, typically for recovering lost bits.
- Interleaving and interpolation: This amounts to making an educated guess about the lost audio. It requires complicated data and signal processing and is MIP-intensive in the decoder.

Our packet-loss recovery scheme is based on transmitting information about more than one audio frame in each IP packet. Referring to Fig. 10, we use the fine-grain scalability of the bitstream to allow us to simply compute and insert copies of frame 1 into packets 2 and 3 with progressively lower resolution, that is, we use shorter EZK bitstreams to produce a redundant 'sub-copy'.

If packet 1 is lost, it is possible to reconstruct a lower quality version of frame 1 from the sub-copy in packet 2. If packets 1 and 2 are lost, we can recover frames 1 and 2 from the lower-resolution sub-copies in packet 3. The decoder buffers incoming packets and reconstructs each frame from the highest-resolution copy that remains.

Because of the flexibility of EZK bitstreams, we have freedom to decide how each packet is segmented between current and previous frames. Thus, we can devote more of each packet to sub-copies when loss rates are high and less when network conditions are favourable. We call this process 'Internet Modulation'.

This recovery process lowers the effective loss rate. For instance, if each packet carries copies of the three previous audio frames, when the Internet loss rate is 10%, the effective rate is 0.01% [19].

We have carried out informal listening tests using total bitrates of 64 and 128 kbps, up to four previous audio frame sub-copies in a packet, and random packet loss of 50%. These were compared with EZK bitstreams with no redundancy, where lost packets were substituted with silence or with the most recent available packet.

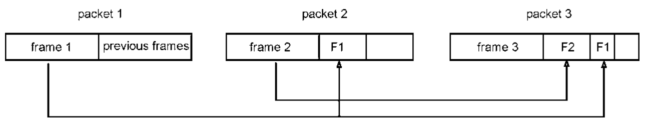

Fig. 10 *Insertion of redundancy to enable recovery from packet loss*

Silence substitution is more objectionable to listeners and the reconstruction is mostly unintelligible. Packet repetition does not provide any advantage either for intelligibility or for SSNR and can introduce unpleasant artefacts when several packets in succession are lost.

But with Internet Modulation, the music listening experience is significantly improved and speech is highly intelligible, except when the loss burst is longer than the four packets of protection. In one example, in 810 frames, this only occurred 27 times, or 3.33% of the time: in other words, an actual loss rate of 50% is transformed to an effective loss rate of ~3%.

3.3 Other advantages

3.3.1 Scalability in the network: In our recent work, we have started to examine other advantages of deploying fine-grain scalable audio codecs for music distribution by streaming. One of the potentially major advantages in fine-grain scalability is that routers can behave more intelligently when faced with mounting congestion. The traditional solution is to discard arriving packets, which led to the development of the Internet Modulation scheme described earlier. Our new alternative [20] involves shaving packets in the routers to attempt to better regulate traffic flows.

Rather than ignore any newly incoming packets until the congestion has diminished, the novel approach we investigated reduces scalable audio packets in size once the buffers in the router exceed a pre-determined capacity limit (prior to buffer overflow). We have examined the behaviour of networks with these kinds of routers against various parameters in Fig. 11. We varied the percentage to drop in each packet, the percentage congestion of the router (100% means all buffers are full), the threshold level of buffer capacity which triggers shaving and the relative amount of scalable audio to background (non-scalable) traffic. The results obtained by simulation were compared against the M/D/1 network model [20].

Preliminary investigations show that the ability to shave packets helps not only the audio traffic but also all the other traffics, because router buffers become less likely to overflow. This is demonstrated in Fig. 11, which plots probability of overflow against buffer occupancy for various shaving thresholds ('R' indicates the remainder or percentage of packet remaining after shaving). The abscissa shows the fullness of a buffer of capacity 30 and the ordinate the percentage probability that the buffer overflows. All the curves shown in Fig. 11 have a knee at ~18, which shows that, for this particular scenario (i.e. shaving threshold = 60%), once packet shaving is initiated, the likelihood of buffer overflow rapidly diminishes, thus preventing packet loss.

We were also able to show that packet loss for both the audio and non-audio (i.e. non-shavable) streams were reduced. Furthermore, both peak and average delays on packet transmission were lower and the variability of packet delay was also reduced (i.e. less jitter). In informal listening tests made on real audio streams processed through the simulator, the user experience is greatly enhanced with packet shaving.

Obviously, there is a small computational cost in the routers, but it is modest because no signal processing is required, only deletion of the end of a bitstream, which is an entirely practical proposition in future generation routers. To do the same on an MP3 stream would require DSP-intensive transcoding and is not likely to be practicable in routers for some time to come.

3.3.2 Layering in IPv6 and future generation packet networks: In present-day Internet, all packets are treated equally by routers, regardless of their importance. Thus, packets for a non-time-critical file transfer are as likely to be discarded as those for streaming audio. Future generation Internet will allow differentiation by means of quality-of-service (QoS) controls. These allow for guaranteed and best-effort service levels.

The best way for streaming media to use these controls is to layer the signal so that a basic version of the content can be transmitted using a guaranteed service, and one or more enhancement layers, which improve the quality, are transmitted by best effort. It is straightforward to produce layers from an EZK bitstream frame, as shown in Fig. 12. The base layer is effectively just the head of the bitstream, the next layer is the middle and the third layer is the tail. Of course, the quality of each layer can be varied with user preference or prevailing network conditions, which is not the case if course-grain scalability of Fig. 9 provides the layered representation. As before, EZK bitstream restructuring comes after the encoding and is completely flexible.

4 Conclusions

In this paper, we have described a novel, fine-grain scalable codec and how it offers functionality appropriate to the Internet. In listening tests, as well as objectively, this codec performs as well or better than the ever-popular MP3 codec. Additionally, its bitrate scalability provides for TCP-friendly protocols, meaning that audio streams cohabit

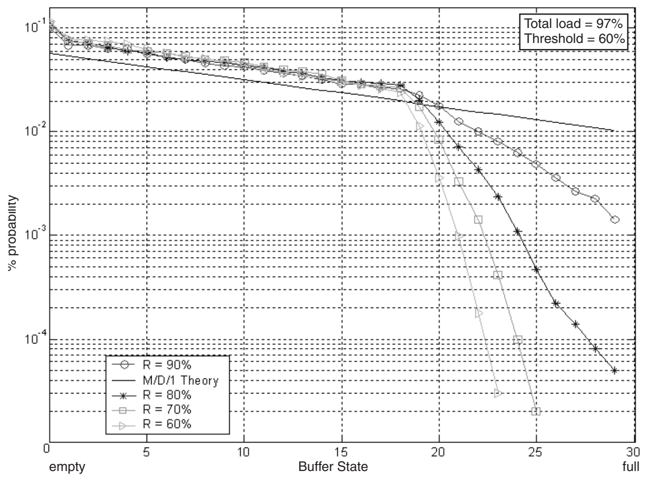

Fig. 11 *Plot showing the effects of packet shaving*

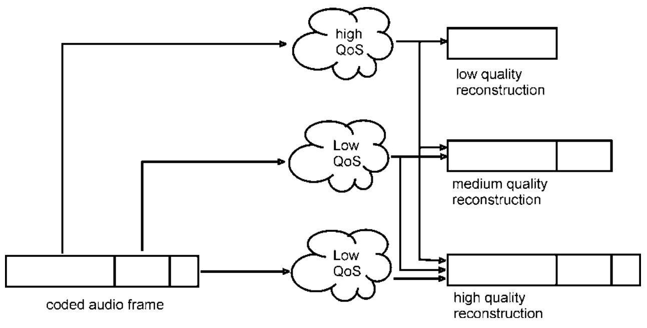
high QoS
Low QoS
Low QoS
low quality reconstruction
medium quality reconstruction
high quality reconstruction
coded audio frame

Fig. 12 *Layering for QoS networks with EZK encoding*

with other data streams, and even offers the potential to alleviate congestion. The fully embedded nature of the coding scheme also has very low computational requirements at encoder, server and client and reduces storage needs because only one file is needed for every bitrate from fully lossless to only a few kilobits per second and less.

This paper also describes a packet-loss repair technique that profitably benefits from the fine-grain scalability. This also has low computational overhead at server and client and does not necessarily increase network traffic. It is designed to cope well with burst loss and leads to improved perceptual quality of the reconstructed signal under packet-loss conditions.

These advantages support mobile and wireless broadband IP in a variety of ways. For instance, as the number of users in a cell increases, the streamed rate to each can be adjusted to best match the total capacity. As a user roams between cells, the packet-loss protection can assist the network infrastructure in providing intelligent handover. And within a cell, as the channel conditions modulate because of multi-path and other transmission issues, the bitrate and loss protection can be modified to cope and will continue to provide the user with the optimal service provision for the prevailing conditions.

From a content provider's perspective, creation and management of coded content is simplified, as there is a single copy of each 'essence' file, which will provide the full range of quality from totally lossless to very low bitrates. As we have also mentioned, scalable streaming has benefits for other traffics, because buffer overflow and therefore packet loss are ameliorated.

Thus, scalable coding of audio offers many benefits to all the players in the entire chain from content provider through service provider and network provider, all the way to the consumer, whose listening pleasure underpins the economics of music delivery.

5 Acknowledgments

The authors would like to acknowledge the contributions to the work underpinning this paper from the following: Ben Leslie and Chris Dunn. They also acknowledge the support from EPRSC, JISC, BT and Insonify. They would also like to thank Professor Jonathan Pitts and Mr Salih Hussein for their contribution to the work described in Section 3.3.1 and Fig. 11.

6 References

1 'Mobile 3G telecoms: vision, meet reality', *Economist*, 2 Sept. 2004

2 Brandenburg, K., and Stoll, G.: 'ISO-MPEG-1 audio: a generic standard for coding of high-quality digital audio', *J. Audio Eng. Soc.*, 1994, **42**, (10), pp. 780–792

3 Painter, T., and Spanias, A.: 'Perceptual coding of digital audio', *Proc. IEEE*, 2000, **88**, (4), pp. 451–513

4 Wang, Y., Huang, W., and Korhonen, J.: 'A framework for robust and scalable audio streaming'. 'Proc. ACM Multimedia 2004', New York, October 2004

5 Grill, B.: 'A bit rate scalable perceptual coder for MPEG-4 audio'. 103rd AES Convention, Preprint No. 4620, New York, 1997

6 Zhang, Q., Wang, G., Xiong, Z., Zhou, J., and Zhu, W.: 'Error robust scalable audio streaming over wireless IP networks', *IEEE Trans. Multimedia*, 2004, **6**, (6), pp. 897–909

7 Li, J.: 'Embedded audio coding (EAC) with implicit auditory masking'. ACM Multimedia 2002, Nice, France, Dec 2002

8 Leslie, B., and Sandler, M.: 'Robust coding for the transmission of audio or video signals'. Patent application PCT/GB 00/01649, 2000

9 Shapiro, J.M.: 'Embedded image coding using zerotrees of wavelet coefficients', *IEEE Trans. Signal Process*. 1993, **41**, (12), pp. 3445–3462

10 Said, A., and Pearlman, W.A.: 'A new, fast and efficient image codec based on set partitioning in hierarchical trees', *IEEE Trans. Circuits Syst. Video Technol.*, 1996, **6**, (3), pp. 243–250

11 Srinivasan, P., and Jamieson, L.H.: 'High-quality audio compression using an adaptive wavelet packet decomposition and psychoacoustic modeling', *IEEE Trans. Signal Process.*, 1998, **46**, (4), pp. 1085–1093

12 Karelic, Y., and Malah, D.: 'Compression of high-quality audio signals using adaptive filterbanks and a zero-tree coder'. Proc. 18th IEEE National Convention, Tel-Aviv, Israel, 1995

13 Leslie, B., and Sandler, M.B.: 'Joint source and channel coding for internet audio transmission'. 106th Convention of the Audio Engineering Society, Preprint No. 4932, Munich, May 1999

14 Princen, J., and Bradley, A.: 'Analysis/synthesis filter bank design based on time domain aliasing cancellation', *IEEE Trans. ASSP*, 1986, **ASSP-34**, (5), pp. 1153–1161

15 Yu, R., Ko, C.C., Rahardja, S., and Lin, X.: 'Bit-plane Golomb coding for sources with Laplacian distributions'. Proc. IEEE ICASSP 2003, 2003

16 Leslie, B., Dunn, C., and Sandler, M.: 'Developments with a zerotree audio codec'. Proc. AES 17th International Conference: High-Quality Audio Coding, Florence, 2–5 September 1999

17 Park, S.-H., Kim, Y.-B., Kim, S.-W., and Seo, Y.-S.: 'Multi-layer bit-sliced bit rate scalable audio coding', 103rd AES Convention, Preprint No. 4520, New York, 1997

18 Floyd, S., Handley, M., Padhye, J., and Widmer, J.: 'Equation-based congestion control for unicast applications', available at http://www.aciri.org/tfrc

19 Leslie, B., and Sandler, M.: 'Adaptive loss recovery and adaptive bitrate scalability using a zerotree audio codec'. 107th AES Convention, Preprint No. 5045, New York, September 1999

20 Pitts, J.M., and Schormans, J.A.: 'Introduction to IP and ATM Design and Performance' (John Wiley and Sons, 2001, 2nd edn.)