SPATIALLY DISJOINT SOURCE CHANNEL CODING: TAKING ADVANTAGE OF THE CURRENT DIAL-UP ARCHITECTURE FOR VIDEO OVER THE INTERNET

Guido M. Schuster, Ikhlaq S. Sidhu, Michael S. Borella

Advanced Technologies Research Center
3COM Carrier Systems Business Unit
1800 W. Central Rd. Mt. Prospect, IL 60056
{guido_schuster, ikhlaq_sidhu, mike_borella}@3com.com

ABSTRACT

In this paper we present a new and powerful approach for transmitting video over the Internet via dial-up modem links. Currently, the source and channel coding are both done in the user PC. In fact, joint source channel coding has become quite popular. In strong contrast to this, we propose to spatially separate the source and channel coding so that the source coding is performed in the user PC and the channel coding is performed in the remote access concentrator. This results in significantly improved video quality since the modem connection has a very small packet loss rate compared to the Internet. In addition, the bit rate of the modem connection is fixed and much lower than the potentially available bit rate of the Internet. Clearly, using a channel coding scheme powerful enough to protect against the high packet loss rate of the Internet is not necessary for the modem connection, and in fact, it is a waste of the already very constrained modem connection bandwidth. By moving the channel coding from the user PC to the remote access concentrator, we free up bits on the highly reliable modem link for the source coding, which in turn results in a compressed video of higher quality. In the remote access concentrator we then employ a channel coding scheme which increases the overall bit rate but also makes the real time stream robust to the highly likely packet drops in the Internet.

1. INTRODUCTION

With the introduction of the world wide web, the mass market has discovered the Internet. While in the early years, the instantaneous access to information was enough to keep the masses entertained, there is now a strong trend of expecting the world wide web to "sing and dance." The quality of television is far superior to the current video streaming products available for the Internet. The main reason for this is the large amount of bandwidth (note that we use the term bandwidth synonymously with bit rate, as it is common in the networking community) required to deliver television quality video. The quality of MPEG-1, which requires around 1.5 Mbits/s is often compared to VHS video.

While many corporate users have Internet connections at T1 (1.5 Mbits/s) and higher speeds, the average user at home is still connected to the Internet using a V.34 (up to 33.6 kbits/s) modem or one of the newly released V.90 (up to 56 kbits/s [but currently constrained by the FCC to a maximum of 53 kbits/s] downstream, up to 31.2 kbits/s upstream) modems. In other words, television quality video is not possible through one of these modems. Clearly there is a growing population which has access to cable modems (around 1.5 Mbits/s downstream, around 300 kbits/s upstream) or an ADSL modem (around 3 Mbits/s downstream, around 1 Mbits/s upstream). Furthermore there are sub-rate DSL initiatives (around 500 kbits/s downstream, around 100 kbits/s upstream) and obviously BRI ISDN at 128 kbits/s.

Nevertheless, the lower rate solutions have penetrated the market much more quickly, because of cost concerns. Clearly most home users already have a V.34 modem and many will upgrade to V.90 in the coming months. Therefore, for the remainder of this paper, we will concentrate on these two modems as means of accessing the Internet.

The paper is organized as follows: In section 2 we discuss the current architecture for modem access to the Internet. In section 3 we discuss packet based forward error correction schemes and introduce a new scheme, which is based on cross-wise parities. We then propose to use this scheme as a channel coder either between two RACs or, if the modem connection is asymmetric, between the sender RAC and the receiver modem. In section 4 we propose to apply intra-standard transcoding at the sender RAC to achieve a highly robust video stream. The main advantage of this approach is, that it does not require the receiver modem to implement a channel decoder. We then conclude the paper in section 5.

2. DIAL-UP ARCHITECTURE

The current dial-up architecture is shown in Fig.1. The PC is connected through a modem to the remote access concentrator (RAC), which terminates the modem connection. From there, the IP packets are routed into the Internet. The basic idea of the proposed spatially disjoint source channel coding is to move the channel coding function out of the user PC into the RAC. This frees up the capacity used on the low bandwidth modem connection for source coding and hence the resulting video quality is significantly better.

Clearly spatially disjoint source channel coding only works well if the probability of packet loss is much larger

in the Internet than in the modem connection. The probability for packet loss in the Internet directly depends on the level of congestion. While several studies (see [1] and the references in [1]) have been conducted on this subject, it is impossible to give a single, meaningful number as the loss rate. Nevertheless, our own experiments have shown that loss rates in the 5% range are quite common and in the worst case, the network can drop as many as 25% of all packets [2].

Since modems are concerned with bit error rates (BER) and not packet error rates, we need to translate the common BER of V.34 modems to a packet loss rate. Note that we assume that if a bit is damaged in a packet, the entire packet is useless and hence lost for the receiver. Since V.34 modems can run at different speeds, one can trade speed for a lower BER. In [3], it is stated that the speed is selected such that the BER is between 10^{-5} and 10^{-6}. Note that a single 2400 bits/s step can change the BER by an order of magnitude [4]. Because of the fact that re-transmission over the modem link (i.e., using V.42) adds delay, it is recommended to run a V.34 modem which is used for real time transmission just below its top speed, i.e., at a BER of about 10^{-6} [4].

Clearly, the longer the packet, the more likely it is to have a single bit error and hence the more likely it is to have that packet dropped. If we assume the fastest speed available, 33.6 kbits/s, and a video frame rate of 10 frames/second, then an average frame contains N = 3360 bits. Hence the average probability for a packet loss on the modem connection is $1-(1-\mathrm{BER})^N = 1-(1-10^{-6})^{3360}$ = 0.3%. Note that this is at least an order of magnitude better than the observed packet loss rates in the Internet. In other words, the packet loss which results from the modem connection is negligible compared to the packet loss in the Internet.

From the PC point of view, the Internet is a low bandwidth (33.6 kbits/s), high loss (up to 25%) packet switched network, which makes it very hard to deliver acceptable quality video. In the above discussion we have shown that in reality, the path to the Internet is a relatively reliable, low bandwidth link concatenated with a relatively unreliable, high bandwidth link. If we assume that the RAC is cooperating with the PC, then we can employ channel coding schemes, which are optimized for the respective link, instead of having an overall scheme, which must deal with a low bandwidth link having a high loss rate.

3. PACKET BASED FEC

The basic idea behind packet based forward error correction (FEC) schemes is to send redundant packets, so that a lost packet can be recovered at the receiver using this redundant information [5].

One class of packet based FEC is based on systematic (N,K) block codes, where for every K original data packets P parity packets are created. These data and parity packets then define a block of N=K+P packets. For example, at RAC 1, for every three packets (K=3), a new packet (P=1) is generated, which contains the XOR of the previous three packets. In other words, sets of three packets are replaced by a block of four packets (N=K+P), where the first three packets are the original packets, and the fourth packet is a parity packet. Clearly, at RAC 2, we can now recover one packet loss in a block of four packets. Note that this simple channel coding scheme results in a packet stream expansion of 33%, but the additional packets are only sent over the Internet and not over the bitrate constrained modem link.

Further note that the simple parity (XOR) scheme described above is a special case of many other possible XOR based schemes. See [6] for a proposed RTP payload format for generic XOR based FECs.

While the above XOR channel coding scheme is very simple, much more powerful methods are known, such as Reed-Solomon (RS) erasure codes [7]. Such an RS code can always recover up to P lost packets in a block of N packets, regardless of the choices for P and N, which is optimal.

3.1. A simple yet powerful (K+3,K) coder

In this section we introduce an XOR based coder, which is as powerful as an RS coder, but conceptually and computationally much simpler. The main restriction of this coder is that we can only add up to three parity packets per block and hence we can only correct up to three packet losses per block. Our experiments, which are consistent with the findings in [8], have shown that this is usually sufficient for an interactive application.

The proposed scheme is based on three parity packets, a "straight" packet PS, an "up" packet PU and a "down" packet PD. If all three packets are used, then P=3 and hence out of N sent packets, three packets can be lost. Note that any subset of these three parity packets can also be used. Therefore, if two parity packets are used, then two packets can be lost, or if one parity packet is used, then one packet can be lost.

3.1.1. Encoder

The main operation of the encoder is to calculate the XOR of the incoming packets and create the parity packets. The names "straight", "up" and "down" of the parities relates to the way they are calculated. Conceptually, for the parity calculation, the packets are lined up and padded with zero, so that they are all of the same length. We then add the packet length and a sequence number in front of the packet, so the structure of a final packet look like this:

[Sequence Number] [Length] [Data] [Zeros].

Assume the original data packet stream is as follows (byte wise), where, for the purpose of the example, the number of data packets (K) is four:

D0: [001] [004] [105] [203][255]
D1: [234] [230] [124]
D2: [127] [002] [019] [045]
D3: [145] [003] [002] [194]

After the sequence numbers and the packet lengths are appended, the structure looks like this, where we assume that the sequence numbers and the packet lengths are each one byte long.

D0: [000] [005] [001] [004] [105] [203] [255]
D1: [001] [003] [234] [230] [124]
D2: [002] [004] [127] [002] [019] [045]
D3: [003] [004] [145] [003] [002] [194]

For PS we calculate the parity from top to bottom and put the result into the parity packet PS which we then transmit. Note that we calculate the parity byte-wise (could be done bit-wise, which might reduce the length of the longest parity packet, more on this later on) and we include the packet length in the parity calculation, but not the sequence number. Hence from now on, we neglect the sequence number. In the following example we printed one straight line in bold to point out how the PS(0)=[006] byte is calculated.

D0: [**005**] [001] [004] [105] [203] [255]
D1: [**003**] [234] [230] [124]
D2: [**004**] [127] [002] [019] [045]
D3: [**004**] [145] [003] [002] [194]
PS: [**006**] [005] [227] [004] [036] [255]

The other two parities PU and PD are calculated similarly, but instead of calculating the parity in a straight line, as for PS, we calculate these parities using +/- 45 degree lines, where PU is calculated using a +45 degree line and PD using a -45 degree line. We now show how PU is found using the above example, where all bytes used for the PU(3)=[244] byte are printed in bold:

PU: [005] [002] [234] [**244**] [036] [239] [047] [194]
D0: [005] [001] [004] [**105**] [203] [255]
D1: [003] [234] [**230**] [124]
D2: [004] [**127**] [002] [019] [045]
D3: [**004**] [145] [003] [002] [194]

As can be seen from the above example, PU can be larger than the largest data packet (D0). It is easy to show that in the worst case, PU (and also PD) are only K-1 bytes longer (if the XOR is done bit-wise, this reduces to K-1 bits) than the longest data packet, while the length of PS is always the same as that of the longest data packet. Clearly, the calculation of the PD works in a very similar manner using the -45 degree lines.

While the above description is accurate, it requires that all the data packets are buffered. We have implemented a more efficient in-place version, which does not need to store the data packets. Note that the in-place version of the encoder uses only the parity packets as memory and the algorithms computational complexity is linear in the number of bytes and the number of data packets (K). In other words, this is a minimum memory and minimum complexity algorithm, which has almost minimum overhead (recall the K-1 bytes/bits PU and/or PD can be larger than the largest data packet).

3.1.2. Decoder

The decoder needs to handle many different cases. As mentioned before, one, two or three packets out of the N sent packets can be lost, and all of the packets can be recovered from the arrived packets. Clearly if we lose a parity packet, we do not need to recover that. On the other hand, we try to recover every lost data packet. To be able to recover lost data packets, we need to receive as many parity packets as there were lost data packets in a given block.

The recovery algorithm can be most easily explained by means of an example. Nevertheless we would like to point out that we have a formally proven that with the presented scheme, the recovery of $i, i \leq 3$ lost data packets is always possible, as long as we have received at least i parity packets for a given block.

If only one data packet is lost, then the recovery is quite simple. This can be seen from the example below, where we assume we lost D1 but PU arrived.

PU: [005] [002] [234] [**244**] [036] [239] [047] [194]
D0: [005] [001] [004] [**105**] [203] [255]
D1: [???] [???] [**???**] [???] [???] [???] [???]
D2: [004] [**127**] [002] [019] [045]
D3: [**004**] [145] [003] [002] [194]

The bold line indicates how a particular byte in D1 is recovered. By calculating the XOR sum of the arrived packets and PU we can recover the D1 byte which is: [004] xor [127] xor [105] xor [244] = [230]. Clearly we do this for every byte, which results in:

D1: [003] [234] [230] [124] [000] [000] [000].

Stripping and reading the first Byte D1(0)=[003] gives us the length. We then use this length to strip off the trailing zeros and hence the final recovered data packet is:

D1: [234] [230] [124].

As we have shown, it is straightforward to recover a single lost packet. The case when two packets are lost is a bit more complicated, and again we use an example to demonstrate how the algorithm works. In the example below we assume that D0 and D2 are lost and PS and PU arrived.

PU: [005] [002] [234] [244] [036] [239] [047] [194]
D0: [???] [???] [???] [???] [???] [???]
D1: [003] [234] [230] [124]
D2: [???] [???] [???] [???] [???] [???]
D3: [004] [145] [003] [002] [194]
PS: [006] [005] [227] [004] [036] [255]

D0(0) can be immediately recovered using PU(0), hence D0(0)=PU(0)=[005]. Then D2(0) can be recovered as follows: D2(0) = PS(0) xor D0(0) xor D1(0) xor D3(0) = [004]. Hence the situation at hand is now as follows:

PU: [005] [002] [234] [244] [036] [239] [047] [194]
D0: [005] [???] [???] [???] [???] [???]
D1: [003] [234] [230] [124]
D2: [004] [???] [???] [???] [???] [???]
D3: [004] [145] [003] [002] [194]
PS: [003] [005] [227] [004] [036] [255]

Note that the first row has been recovered completely. This two step recovering can now be applied row by row, which results in the following picture:

D0: [005] [001] [004] [105] [203] [255]
D1: [003] [234] [230] [124]
D2: [004] [127] [002] [019] [045] [000]
D3: [004] [145] [003] [002] [194]

Again, stripping and using the length information, results in the two recovered packets D0 and D2:

D0: [001] [004] [105] [203] [255]
D2: [127] [002] [019] [045].

The last case we need to address is when three data packets are lost and all three parity packets are available. Assume D0, D1 and D3 are lost:

PU: [005] [002] [234] [244] [036] [239] [047] [194]
D0: [???] [???] [???] [???] [???] [???]
D1: [???] [???] [???] [???] [???] [???]
D2: [004] [127] [002] [019] [045]
D3: [???] [???] [???] [???] [???] [???]

PD: [004] [149] [127] [239] [054] [085] [105] [203] [255]

PS: [006] [005] [227] [004] [036] [255]

Again, D0(0) =PU(0)=[005]. Also, D3(0)=PD(0)=[004]. Hence D1(0)=PS(0) xor D0(0) xor D2(0) xor D3(0) = [003]. Hence the situation at hand is:

PU: [005] [002] [234] [244] [036] [239] [047] [194]
D0: [005] [???] [???] [???] [???] [???]
D1: [003] [???] [???] [???] [???] [???]
D2: [004] [127] [002] [019] [045]
D3: [004] [???] [???] [???] [???] [???]
PD: [004] [149] [127] [239] [054] [085] [105] [203] [255]

PS: [006] [005] [227] [004] [036] [255]

Note that the first row has been recovered completely. This three step recovering can now be applied row by row, which results in the following picture:

D0: [005] [001] [004] [105] [203] [255]
D1: [003] [234] [230] [124] [000] [000]
D2: [004] [127] [002] [019] [045]
D3: [004] [145] [003] [002] [194] [000]

Again, stripping and using the length information, results in the two recovered packets D0, D1 and D2:

D0: [001] [004] [105] [203] [255]
D1: [234] [230] [124]
D3: [145] [003] [002] [194].

While the above description is accurate, it requires that all the data packets are buffered. We have implemented a more efficient in-place scheme, which does not need to store the data packets. Note that the in-place implementation of the decoder uses only the parity packets as memory and the algorithms computational complexity is linear in the number of bytes and the number of data packets (K). In other words, this is also a minimum memory and minimum complexity algorithm.

3.2. Forward error correction between two RACs

As pointed out in section 2, the channel coding should only be applied for the Internet. This can be achieved by using a packet based FEC between RAC 1 and RAC 2. In other words, a packet based FEC encoder is applied to the packet stream in RAC 1 and a packet based FEC decoder is applied to the received packet stream in RAC 2.

3.3. Forward error correction between a RAC and a receiver PC

The new V.90 modems are asymmetric, in other words, the downstream (Internet to user) direction can carry data at speeds up to 56 kbits/s, while the upstream (user to Internet) is constrained to a maximum speed of 31.2 kbits/s. Therefore, in an end-to-end connection between two V.90 modems (modem-Internet-modem) almost half of the bit rate is unused, since the modem cannot send at the same high speed it can receive. We propose to use this available downstream bit rate by passing the parity packets along to the receiver modem 2, instead of using them to reconstruct the missing packets in RAC 2. This has two major advantages. First, the processing load is distributed from the RAC to the modem/PC and second, if packets are lost in the receiver modem link, they can still be recovered.

In the case that the end-to-end connection is not a modem-Internet-modem connection, but for example as seen in Fig. 1, a modem-Internet-T1 connection to a corporate video server, one side of the FEC calculation can again be off-loaded from RAC 2 to the server, in this case the video database.

4. ERROR CONCEALMENT BY TRANSCODING IN THE RAC

While the above schemes require the receiver RAC 2 or the receiver PC 2 (or video server) to be aware of the FEC scheme, in this scheme, we apply transcoding in the sender RAC 1. Transcoding takes a bit stream of a certain video standard and changes it into another video standard. For our application, the second bit stream will use more bandwidth but be less susceptible to a lossy network. Furthermore, for our application, the second bitstream still represents the same video standard, but it uses a different set of parameters. For the current implementation, we use H.263 as the video compression standard, because of its popularity for very low bit rate applications.

Like all other efficient video coders, H.263 uses motion compensated prediction as shown in Fig. 2. As can be seen in Fig. 2, the entropy coded (EC) quantized discrete cosine transform (DCT) coefficients of the displaced frame difference (DFD), and the EC differential pulse code modulated (DPCM) displacement vector field $\vec{d}_k$, are sent to the decoder. In other words, we send the prediction error and the motion information. While motion compensated prediction is a very efficient way of reducing the overall bit rate for a given quality, it is also very sensitive to lost data.

To illustrate this point, imagine that a packet is lost, which carries the motion information $\vec{d}_k$ for the current frame. Hence the receiver can only use the prediction error and an estimate of the motion to generate the frame at the receiver. Hence this frame is incorrect and from now on, the encoder and the decoder use different reference frames for the encoding and decoding. This results in visible errors in the received video stream. Because of this and other reasons, the encoder sends a so called intra frame every few seconds, so that the encoder and decoder are synchronized. In summary, we need motion compensated video compression to achieve acceptable video quality for the highly constrained bandwidth of the modem link, but motion compensated prediction is very sensitive to lost data packets, which are likely in the Internet.

The solution we propose is based on intra standard transcoding of the H.263 stream in RAC 1. In the case where an interactive video telephony session between two V.90 modems is underway, we again have more bandwidth available in the downstream direction, than in the upstream direction. Hence we propose to decode the H.263 video stream in the RAC 1 and re-encode it not using motion information, but only using intra block encoding and conditional replenishment. While intra block encoding is simply a macro block encoded independent of anything else, conditional replenishment works as follows. If the motion is considered large enough (we already have the motion vector for this block), then the block is sent as an intra block. If the motion is not large enough, then we do not change the

block, i.e., in H.263 the COD bit is set to 0. Clearly this will result in a new H.263 stream representing the same original video sequence, but this time, no motion information is used. Hence the bit rate increases significantly. Since the bit rate is constrained to 56 kbits/s for a V.90 modem, we need to use some form of rate control. A simple approach is based on changing the threshold for the conditional replenishment. This does though results in a jerky sequence. Hence we use a rate control which adjust the quantizer step size, so that the overall target bit rate can be achieved. Note that we only use conditional replenishment for macro blocks which do not carry any information. In other words, we keep the settings of the COD used in the highly compressed H.263 bitstream.

In summary, in the RAC 1, we receive the highly compressed H.263 stream, decode it, encode it again not allowing any motion compensation (i.e., no inter blocks) and adjust the quantizer step sizes so that the new downstream target bit rate (56 kbits/s) can be achieved. The resulting compressed video stream looks slightly more blocky, because of the larger quantizer step size applied, but otherwise it is basically identical with the original stream from a visual point of view. On the other hand, this stream is now extremely robust to packet loss in the Internet and it does not require any FEC decoding on the receiver side. In fact, the receiver RAC 2 and/or the receiver PC 2, do not need to know that this was done in the RAC 1. Also note, that the computational requirements are rather small, since for the encoding process, no motion estimation is needed, since the motion vectors are already present in the highly compressed stream. Further note that if PB frames are used, we do not change the B frames, since they are never used as reference frames for prediction.

5. CONCLUSIONS

In this paper we proposed a new approach to video over the Internet which takes advantage of the current dial-up architecture. We moved the channel coding out of the PC and into the RAC, since the modem connection is quite reliable but severely bandlimited. We presented two approaches, packet based FEC between RACs and transcoding in RAC 1. The main advantage of the former is the ease of implementation, while the main advantage of the later is that only the sender RAC needs to cooperate with the sender PC. We have run experiments for all the above schemes, assuming 25% uniform packet loss and a V.34 connection at 28.8 kbits/s. The results clearly demonstrate that even under these extreme loss conditions, the proposed schemes still result in acceptable video quality, while the traditional schemes do not. In particular, if no FEC is used, then any packet loss results in very low picture quality. If FEC is used, but it is employed in the PC instead of in the RAC, the available bandwidth for source coding is so low, that the resulting video quality is not acceptable.

6. REFERENCES

[1] M. S. Borella, D. Swider, S. Uludag, and G. Brewster, "Internet packet loss: Measurement and implications for end-to-end QoS," in *International Conference on Parallel Processing*, (Minneapolis, Minnesota), Aug. 1998.

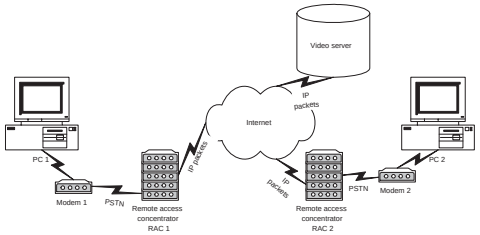

Figure 1: The current dial-up architecture.

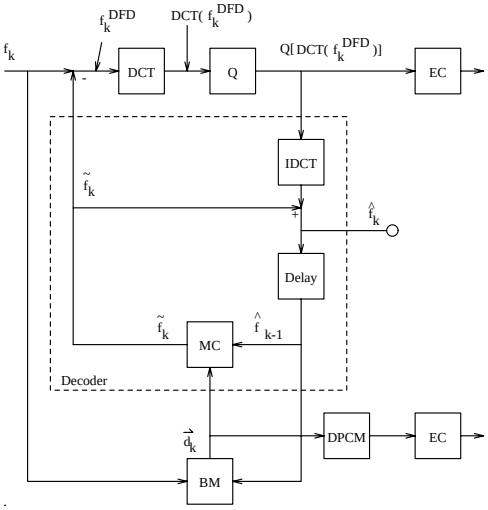

Figure 2: Block diagram of H.263.

[2] T. Kostas, M. Borella, I. Sidhu, G. Schuster, J. Grabiec, and J. Mahler, "Real-time voice over packet switched networks," *IEEE Network magazine*, vol. 12, pp. 18–27, Feb. 1998.

[3] G. D. Forney Jr., L. Brown, M. V. Eyuboglu, and J. L Moran III, "The V.34 high-speed modem standard," *IEEE Communications magazine*, vol. 34, pp. 28–33, Dec. 1996.

[4] D. Lindbergh, "The H.324 multimedia communication standard," *IEEE Communications magazine*, vol. 34, pp. 46–51, Dec. 1996.

[5] C. Perkins and O. Hodson, "Options for repair of streaming media, draft-ietf-avt-info-repair-03." Internet-draft, Work in progress, Mar. 1998.

[6] J. Rosenberg and H. Schulzrinne, "An RTP payload format for generic forward error correction, draft-ietf-avt-fec-02." Internet-draft, Work in progress, Mar. 1998.

[7] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM SIGCOMM Computer Communication Review*, pp. 24–36, Apr. 1997.

[8] J. Bolot and A. Vega-Garcia, "The case for FEC based error control for packet audio in the Internet," *ACM Multimedia Systems*. to appear.