REAL-TIME ADAPTIVE FORWARD ERROR CORRECTION FOR MPEG-2 VIDEO COMMUNICATIONS OVER RTP NETWORKS

Bulent Cavusoglu, Dan Schonfeld*, and Rashid Ansari

University of Illinois at Chicago
851 S. Morgan St. (M/C 154)
Chicago, IL 60607-7053
*Tel: +1 312 996-5847
{bcavusog, ds, ansari}@ece.uic.edu

ABSTRACT

We present an algorithm for real-time adaptive forward error correction (FEC) of MPEG-2 video stream, encapsulated using real-time transport protocol (RTP) and delivered over best-effort networks. Our algorithm provides an efficient method to determine the allocation of redundancy to the MPEG-2 video stream. The redundancy is allocated such that the resulting estimated degradation density function for video (DDF) is uniformly distributed. A weight, which indicates the relative importance of RTP packets, together with the communication channel characteristics and FEC scheme are used to model the density function of the video stream and allow us to determine the allocation of FEC packets. The weight is based on the content of RTP packets in the video stream. Parameters extracted from the RTP header are used to determine the weights, so that the proposed algorithm can be implemented in real-time. In our simulations, we have relied on motion compensation and group of picture (GOP) data to determine the relative weights. Simulation results provided established the significant improvement in performance based on our proposed approach to adaptive FEC.

1. INTRODUCTION

Real-time video applications have very strict time constraints for the delivery of video content. If a packet is delivered to its destination beyond those constraints it is considered as a lost packet. Real-time video applications are therefore built over UDP, where packets are delivered only once unlike TCP where transmission is repeated until the packet is successfully transmitted. Since UDP does not provide any error corrections, recovery from errors has to be achieved by other layers. Packetized networks are said to encounter losses in packets, not in number of message packets and n-k is the number of redundant packets, message packets are fully recoverable if and only if at least any k packets out of n packets arrive at the destination [1,2].

Many effective FEC codes, such as Reed Solomon, Hamming and CRC, [3] have been proposed to recover packet losses. These codes have erasure correcting capabilities as well. FEC codes introduce redundant packets in order to recover lost packets. Increasing the amount of redundancy will achieve higher recovery rates, however it is also evident that increasing the redundancy abruptly will cause congestion and eventually degradation of video quality. The amount of redundancy can be determined by using various performance parameters and constraints, such as required QoS, delay, packet loss ratio and/or network conditions. Moreover once the allowable, optimized or predetermined, extent of redundancy is known, it should be used in the best way.

Previous studies focused on adaptively changing the number of redundant FEC packets depending on QoS requirements [4,5,6]. Particularly, previous efforts have demonstrated the value of adaptive FEC based on picture types [6]. In [7] Adaptive FEC is applied by considering the effects of lost packets on spatial distortion. These studies motivated us to further analyze the structure of MPEG-2 video in real-time to propose an algorithm based on relative visual degradation probabilities.

We will briefly outline some relevant properties of MPEG-2. MPEG-2 video uses spatial and temporal dependencies in a video stream to get the desired compression. MPEG-2 video is constructed from video sequences, which in general has GOP (group of pictures) structures including combination of I (intra-coded), P (predictive-coded) and B (bi-directionally predictive-coded) pictures. Each picture is a made up of slices and each slice is made up of macroblocks. I pictures are constructed with only intra macroblocks, whereas P and B pictures include both intra and predicted macroblocks. P picture uses previous most recent I or P pictures and B picture uses previous or future I or P pictures as references (see Figure 1). As we can follow easily from figure 1, there is either a direct or an indirect dependency from the last picture to the first picture in an MPEG-2 video stream until a new I picture is introduced. Hence, losing one packet from a picture may cause a disturbing visual degradation due to propagation of error, where there is temporal dependency among the pictures. MPEG-2 uses motion vectors to point to the new location of macroblocks in motion (see Figure 2). We will refer to the parts of MPEG-2 encoding where motion vectors are used as “captured motion” to simply identify that some of the motion in the video stream might be intra coded by MPEG-2 encoder because it may become costly to use motion compensation or the required motion vector may exceed the allowed range of the motion vector. Real-time transport protocol (RTP) describes an encapsulation scheme for real-time applications. RTP adds the necessary timing information to its header for real-time applications. RTP also includes MPEG-1 and MPEG-2 video header and MPEG-2 specific video header extensions (see Figure 3) to allow error recovery applications to build fast recovery methods by using the information provided on the header fields. We use RTP header and MPEG-2 specific header extension to gather the vital information of MPEG-2 video stream and build our algorithm.
based on the parameters in the header. This will allow us to run our algorithm in real-time as RTP packets become ready to be sent. In the remainder of the paper we will only explain the parameters we use in our algorithm. The interested reader is referred to [8] for the detailed explanation of MPEG-2 payload for RTP and to [9] for the generic FEC method used with RTP.

Figure 1. Pictures in a typical GOP and temporal dependencies.

Figure 2. Captured motion between pictures.

2. ADAPTIVE FEC

We have constructed our model based on RTP encapsulated MPEG-2 and FEC packets (see Figure 4). It is assumed that packet losses are uniformly distributed and that errors do not occur in bursts. We chose the parameters for simplicity in our model and we show that our algorithm takes the error correction capabilities and channel characteristics into account in its decision criteria. Thus, the algorithm can easily be adapted for different network models as well.

The suggested adaptive FEC algorithm adopts the idea that every picture has a different level of importance in MPEG-2 video. When determining the level of importance for each picture, it is crucial that the parameters, which the algorithm uses, must be reachable in real-time. Determining the level of importance by decoding the MPEG-2 video is not desirable since it is a time consuming process. Thus, we use the header fields of RTP packets to gather the necessary information. In this algorithm, we will be using picture type parameter ‘P’ and ‘f_code’ parameters to determine the weights of each picture. F Code is used in motion vector decoding and it is a measure of how big the motion vector is (see Table 1).

<table>
<thead>
<tr>
<th>f_code(<em>,</em>)</th>
<th>Vertical components of field vectors in picture pictures</th>
<th>All other cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(forbidden)</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>[-4:+3.5]</td>
<td>[-8:+7.5]</td>
</tr>
<tr>
<td>2</td>
<td>[-8:+7.5]</td>
<td>[-16:+15.5]</td>
</tr>
<tr>
<td>3</td>
<td>[-16:+15.5]</td>
<td>[-32:+31.5]</td>
</tr>
<tr>
<td>4</td>
<td>[-32:+31.5]</td>
<td>[-64:+63.5]</td>
</tr>
<tr>
<td>5</td>
<td>[-64:+63.5]</td>
<td>[-128:+127.5]</td>
</tr>
<tr>
<td>6</td>
<td>[-128:+127.5]</td>
<td>[-256:+255.5]</td>
</tr>
<tr>
<td>7</td>
<td>[-256:+255.5]</td>
<td>[-512:+511.5]</td>
</tr>
<tr>
<td>8</td>
<td>[-512:+511.5]</td>
<td>[-1024:+1023.5]</td>
</tr>
<tr>
<td>9</td>
<td>[-1024:+1023.5]</td>
<td>[-2048:+2047.5]</td>
</tr>
<tr>
<td>10-14</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>(used when a particular f_code will not be used)</td>
<td></td>
</tr>
</tbody>
</table>

Table 1. f_codes.
Figure 5. Range of the motion vector.

Let us consider that we have total M pictures per GOP,

$$M = (N_p + 1) \cdot (N_B + 1)$$  \hspace{1cm} (1)

Where $N_p$ is the number of P pictures per GOP and $N_B$ is the number of consecutive B pictures between P pictures.

Then initial weights are given by,

$$w_i = M , \quad w_p = (M - T_p) , \quad w_B = 1$$  \hspace{1cm} (2)

where $w_i, w_p, w_B$ are the initial weights for I, P, B pictures respectively. $T_p$ is the temporal position of the P picture in the current GOP. For instance if arrived picture is the first P picture after I picture then $T_p = N_B + 1$. For the sake of simplicity, it is assumed that each lost macroblock will cause the same amount of distortion in the current picture regardless of its position. The initial weights can be estimated by assuming any error occurring in the reference picture will result in the same amount of additional error in the pictures that use the reference picture. This happens if the macroblock in error is exactly copied from the reference picture. However, copying is hardly the case when motion is present. Next, we will determine the amount of temporal dependency between the pictures and adjust the weights accordingly. Let us define

$$I_{pp} = R_p / R_i , \quad I_{BB} = R_B / R_i$$  \hspace{1cm} (3)

where $I_{pp}$ and $I_{BB}$ are called the intra refreshing factors for P and B pictures respectively, and $R_i, R_p$ and $R_B$ are the sizes, in bits, of compressed I, P and B pictures respectively. If every picture in GOP were coded without the use of motion compensation, then the size of each picture will be approximately equal to each other assuming there is no scene change in GOP (It is possible for an MPEG-2 encoder to detect a scene change and force an I picture wherever there is a scene change). Intra refreshing factors will give us approximately the percentage of intra coded macroblocks in the picture. There is some overhead ignored in this calculation, which is due to motion vector coding and DCT coefficients of residual macroblocks, however we found throughout our simulations that ignoring the overhead does not have significant effects on the results. We now define the maximum possible captured motion amount by using f-codes. Table 1 depicts the motion vector ranges for f-codes. Then X and Y (see figure 5) are given by,

$$X(*) = (2(f\_code(*,0)-1))/2$$

$$Y(*) = (2(f\_code(*,1)-1))/2$$

$$MV_f = \sqrt{x(0)^2+y(0)^2} , \quad MV_B = \sqrt{x(1)^2+y(1)^2}$$

$$MV = \max(MV_f,MV_B)$$

where $MV_f$ and $MV_B$ are forward and backward maximum captured motion amounts respectively and MV is the maximum captured motion amount. MV is a measure of possible amount of motion of an object that could be encoded using motion vectors. However, this should not be confused with the actual motion between the pictures, since some of the motions can be coded “intra” for various reasons. Basically, MV provides the amount of possible dependency between the current frame and the reference frame. In case of a loss in the reference frame concealment techniques will give worse results than that in the case of smaller MV. This is due to the fact that spatial correlation between neighboring macroblocks is less than that in the case of smaller MV.

Here MV and $I_B$ provide the correlation between the current picture and the reference picture. Recall that the initial weights are defined by assuming that every macroblock is copied to the current picture from reference picture. In other words, it is assumed pictures in GOP have correlation 1. A correlation measure between pictures is proposed according to the following formula

$$C_s = 1 - (I_{pp} + \lambda \cdot (1-I_{rr}) \cdot MV / \max(MV))$$  \hspace{1cm} (6)

where $\lambda$ is the adjustment factor. $\lambda$ can take different values for different streams, the optimized value for $\lambda$ might be different for different types of scenes. $\lambda$ is determined by taking into consideration the spatial correlation in the picture. We have determined, throughout our simulations, that taking $\lambda = 1$ gives good results on average. The weights for the current pictures can then be calculated by,

$$W_i = w_i \cdot (1-C_s) , \quad W_p = w_p \cdot (1-C_s) , \quad W_B = w_B \cdot (1-C_s)$$  \hspace{1cm} (7)

Notice that the higher the correlation between the reference picture and current picture is the less the weight for the current picture is. In the extreme case correlation will be 1, which indicates that receiving only the reference frame will be sufficient to recover the current frame, therefore current frame does not need any protection. However, this formulation is not complete since the initial values are determined by assuming a correlation factor of 1. The true correlation values have to be fed back to the previous pictures in GOP and the initial values have to be updated. It is necessary to wait until the end of GOP to determine the weights exhaustively for all pictures in GOP. However this will be impossible to realize in real time because it would be too late to send the FEC packets by the time the actual weights are determined. In this case we update the initial values by waiting only until next P picture. There is no extra delay introduced by doing so because in the encoding order P pictures come immediately after the I picture. The feedback values are given by the following,
Then finalized weights are,

\[ W'_f = W_f \cdot f_f, \quad W'_p = W_p \cdot f_p, \quad W'_h = W_h \]

(9)

We now can define the probability of not recovering a lost packet and then establish the final product for the degradation density function. The probability of not recovering a lost packet in an \((n, k)\) erasure code is simply given by,

\[
P_L = \sum_{j=k+1}^{n} \binom{n}{j} \cdot (1-PLR)^{j-i} 
+ \left[ \sum_{i=0}^{k} \binom{n}{i} \cdot PLR \cdot (1-PLR)^{i-j} \sum_{j=k}^{n} \binom{n-k}{j} \cdot PLR^j \cdot (1-PLR)^{n-j} \right]
\]

(10)

where \(PLR\) is the packet loss ratio. Then the resulting degradation density function is given by,

\[
DDF = P_L \cdot W'_{f}
\]

(11)

Once DDF is obtained, it has to be set to a constant, preferably to an optimized value. Solving the optimization problem is beyond the scope of this paper. Instead, we are going to determine a working point and we will explain how it can be determined in the following section.

3. SIMULATIONS AND DISCUSSION

We use the network model in figure 4 and we also set our FEC code to \((n, n-1)\). In other words, we produced only one FEC packet per \((n-1)\) RTP packet. After generating the RTP packets, we applied our algorithm by assuming that the amount of redundant packets to be added is given. Given the allocated amount of FEC, we set an initial threshold value, \(DDF_0\), for DDF. Whenever a packet arrives DDF is calculated and added together. If the total is less than \(DDF_0\), then received packets are stored. Once the total is greater than \(DDF_0\), one FEC packet is produced with the packets received by simply applying XOR operation among them. At the end of each GOP the percentage of FEC packets generated are compared with the allocated percentage and threshold value is adjusted to be as close as to the percentage allocated. If \(K\) packets are generated per \(M\) packets and the allocated percentage is \(A\%\) then new threshold value is set to

\[
DDF_0 = DDF_0 \cdot A \cdot (M/K)
\]

(12)

We applied our algorithm to MPEG-2 streams, which we generated, with scene detections. We have used scenes from the movies Matrix and Mummy to generate multiple MPEG-2 streams. For different \(\lambda\) values the results were different. The range for \(\lambda\) is between 0 and 1. The best results were obtained when \(\lambda = 1\). We are presenting the average behavior of all simulations, when \(\lambda = 1\). In our simulations we used 25% FEC redundancy amount and we compared the performances of static FEC, where redundant packets are distributed equally, static IPBFEC, where the only criteria used for distribution of FEC packets was the picture type, and our adaptive algorithm, AFEC. Then for performance comparison we measured end-to-end PSNR values. Figure 6 shows the end-to-end PSNR values for our proposed algorithm AFEC (Adaptive FEC), static FEC and static IPBFEC. It can easily be seen that AFEC outperforms the other two methods used for the distribution of FEC.

4. REFERENCES


