October 2021

SUPER FORWARD ERROR CORRECTION (FEC) FOR LONG-TERM REFERENCE FRAME (LTRF) RECOVERY FRAMES

Thomas Davies

SUPER FORWARD ERROR CORRECTION (FEC) FOR LONG-TERM REFERENCE FRAME (LTRF) RECOVERY FRAMES

AUTHORS:
Thomas Davies

ABSTRACT

A Forward Erasure Correction (FEC) scheme improves resilience but can be overwhelmed by losses. Using long-term reference frames (LTRFs), the prediction structures in a video coding may be reset at a special recovery frame. Consequently, there is a need to adequately protect LTRFs. To address such a need, techniques are presented herein that support a FEC scheme for protecting LTRFs without increasing latency. Aspects of the techniques presented herein treat the LTRFs as a base layer for the purposes of layered FEC coding. Thus, it is possible to think of a normal scalable coding scheme with, for example, two temporal layers as in fact consisting of three layers for FEC purposes. Further aspects of the techniques presented herein support a sliding window or convolutional code which offers the opportunity to reduce delay by using a stateful FEC process where the packet memory is incremented by the latest packet before a new packet is produced. Alternatively, a special strong code may instead be used for the LTRFs which depends upon (1) the previous LRTFs and (2) the most recent Layer 0 frames such that if either set is uncorrupted the LTRF can be recovered.

DETAILED DESCRIPTION

A Forward Erasure Correction (FEC) scheme improves resilience but can be overwhelmed by losses. Using long-term reference frames (LTRFs), the prediction structures in a video coding may be reset at a special recovery frame, but this can only be performed if the recovery frames are correctly received. In some instances, such frames may not be correctly received in periods of loss. When such frames are not received, an instantaneous decoding refresh (IDR) frame must be sent instead. Therefore, adequately protecting LTRFs is important.

It is well known that LTRFs can be used to aid error recovery by allowing a current frame to be predicted from old data that has been correctly received, especially if there is a feedback channel that allows a transmitter to know when such recovery data has been correctly received. It is also well known that a FEC scheme may be used to protect data on a link. However, these two processes are usually considered separately in real systems. Recovery may therefore fail since the prior LTRFs may themselves be corrupted or the recovery frame (usually also an LRTF) is either lost or corrupted.

Aspects of the techniques presented herein treat the LTRFs as a base layer for the purposes of layered FEC coding. In this way, it is possible to think of a normal scalable coding scheme with, for example, two temporal layers as in fact consisting of three layers for FEC purposes.

The typical layered FEC scheme is well-known. At each layer, the FEC packets for that layer protect both that layer and the lower (more important) layers. For purposes of illustration, a two-layer system would comprise Layer 0 picture data and Layer 1 picture data that could alternate. For example:

```
.. P0 P1 P0 P1 P0 P1 P0 P1 ...
```

FEC packets for each layer may be produced and interleaved. For example:

```
.. P0 F0 P1 F1 P0 F0 P1 F1 ...
```

An F0 packet would be formed from P0 packets and an F1 packet would be formed from P1 and P0 packets.

Under aspects of the techniques presented herein, a first important element provides a special FEC code that only applies to Layer L (i.e., the stream of recovery LRTFs within Layer 0). Thus, if a Layer 0 frame is an LTRF then the packets which encode it may be denoted as PL packets. Additionally, there are FL FEC packets which are formed only from PL packets. For example:

```
.. P0 F0 P1 F1 PL FL P0 F0 P1 F1 P0 F0 P1 F1 PL FL ..
```

Normally, more than one input frame packet is required in order to produce an FEC packet output. In a block coding scheme, this means buffering packets from the current and previous frames before an output can be made, which increases latency. However, since LTRFs occur relatively infrequently, perhaps only once per second, a large delay is incurred to produce an FEC code for them when several must be aggregated.

Under aspects of the techniques presented herein a second important element provides a sliding window or convolutional code that offers the opportunity to reduce delay by using a stateful FEC process where the packet memory is incremented by the latest packet before a new packet is produced. Any FEC packet thus produced protects the current packet and any previous packets. If such a code were used, it would not be necessary to include an aggregation delay. Therefore, FEC schemes with very long memory are possible without extra latency.

A variant of the second element, according to aspects of the techniques presented herein, may instead use a special strong code for the LTRFs which depends upon (1) the previous LRTFs and (2) the most recent Layer 0 frames such that if either set is uncorrupted the LTRF can be recovered. In this way, there may well be no need for a return path for LTRF acknowledgement since the code is strong enough that delivery can be assumed. In this case a block code could be possible, since the window for FEC packet generation comprises only a few Layer 0 frames and it need not include previous LTRFs. Consequently, the latency for this code may be acceptable. Additionally, such an approach provides a variable-strength version.

In brief, and as described and illustrated above, aspects of the techniques presented herein employ a variable-strength or sliding window layered FEC scheme in conjunction with LTRFs and a special FEC construction to provide special protection for those frames as if they formed a very low frame rate base layer, without incurring additional latency (and in fact with lower latency than deployed FEC schemes).

In summary, techniques have been presented herein that support a FEC scheme for protecting LTRFs without increasing latency. Aspects of the techniques presented herein treat the LTRFs as a base layer for the purposes of layered FEC coding. Thus, it is possible to think of a normal scalable coding scheme with, for example, two temporal layers as in fact consisting of three layers for FEC purposes. Further aspects of the techniques presented herein support a sliding window or convolutional code which offers the opportunity to reduce delay by using a stateful FEC process where the packet memory is incremented by the latest packet before a new packet is produced. Alternatively, a special strong code may instead be used for the LTRFs, which depends upon (1) the previous

LRTFs and (2) the most recent Layer 0 frames such that if either set is uncorrupted the LTRF can be recovered.