Telecommunications Standardization sector
Study Group 15
Experts Group for Video Coding and Systems
in ATM and Other Network environments

Document # AVC-735
Version Final
January 1995
Japan

SOURCE: AT&T Bell Laboratories (Barry G. Haskell and Amy R. Reibman)
TITLE: Feedback of received jitter for RTD model
PURPOSE: Informational and Proposal

1 Introduction

ISO MPEG [1] is in the process of defining a Real-Time Interface Specification which constrains the real-time byte delivery schedule of Transport Streams to real decoders. The Real-Time Interface Decoder Model (RTD) defines the RTI requirements. It consists of 3 buffers for each elementary video stream: the transport buffer, the multiplexing buffer, and an elementary stream buffer. These buffers will be sized to take into account the fact that bytes in the elementary stream will arrive with some jitter. In the case that the bitstream has been transmitted over an ATM network, this jitter is referred to as cell delay variation (CDV).

There is currently much discussion as to the correct value of CDV that should be incorporated in the RTI and RTD. ISO is requesting that the maximum CDV that a decoder would need to be designed to handle is 50 usec. ITU requests 1 msec, and the ATM Forum requests several msec. For certain LANs, jitter may be 100s of msec. If the actual CDV exceeds the maximum that the system has been designed for, correct buffer operation cannot be assured.

Here, we propose to send information about the CDV, as estimated at the decoder, back to the encoder. This information includes possible additional buffer space that the decoder has that can be used to accommodate the CDV. The encoder then has the option of adjusting its parameters to avoid decoder buffer overflow and underflow. This alleviates the need for the decoder to be designed with enough buffering to prevent overflow and underflow given the maximum possible jitter. It also minimizes the delay.

Herein, we describe two ways in which the encoder can adjust its parameters. First, the encoder assumes a smaller-than-actual decoder buffer and encodes to meet the stricter requirements. Second, the encoder can increase the buffering delay by skipping a frame (in the low-delay mode) when decoder buffer underflow is imminent. It can also decrease the buffering delay as described below. In no case is the Video Buffer Verifier (VBV) violated.

In the following, we distinguish between the actual physical decoder buffer size and the amount of buffer allocated by the VBV size.

2 Background

In [2], we showed that, assuming no delay jitter, the encoder could prevent decoder buffer overflow and underflow by computing the actual decoder buffer fullness. In the presence of jitter, we presented a solution for the decoder to prevent decoder buffer overflow and underflow [3] by incorporating additional delay and additional decoder buffer space. Given guaranteed bounds

on jitter, decoder buffer overflow and underflow could be eliminated by choosing sufficiently large buffers and sufficiently large "jitter delay". This solution can be thought of as having the decoder use its available buffering to "shift the VBV" to ensure no buffer overflow or underflow. However, it requires that the physical decoder buffer be sufficiently large. In practice, the decoder buffer may not have enough room to absorb both the VBV fluctuations and the jitter. In addition, it incurs additional decoding delay, and is practical only for the elementary stream decoder buffer.

An alternative solution, briefly noted in [3], would be to have the video encoder modify its data generation rate to ensure that in the Systems Target Decoder (STD) the buffers never emptied below or filled above some prescribed values determined by the expected jitter.

In [2], the encoder ensured correct operation of the decoder buffers, assuming no delay jitter. However, if delay jitter is present, the encoder cannot know the exact fullness of the decoder buffer. Here, we extend the buffer dynamic equations to include the possibility of short-term jitter, where data arrive either early or late to the decoder. The maximum delay jitter is defined to be J_{max}. We assume the decoder clock is stable and accurate.

The current solution does not increase the decoding delay, but fits the jitter delay within the existing buffering delay. It can be thought of as "shrinking the size of the VBV".

The following analysis uses a discrete time interval T, which can be chosen appropriately depending on the context. We focus on the solution for the elementary video stream decoder buffer, although the same principles apply to both the multiplexing buffer and the transport buffer. In general, both the encoder and multiplexer must be aware of the presense of jitter during their operation.

2.1 Buffer dynamics

Because of arrival jitter, the encoder cannot know exactly the fullness of the decoder buffer. The encoder can, however, compute a nominal decoder buffer fullness that corresponds to the case when there is no jitter. The actual decoder buffer fullness will differ from this because of the jitter.

We assume a buffering delay of LT seconds. (For convenience, we assume L is an integer, although it is not necessary.) Underflow is allowed for the elementary stream decoder buffer only when the low_delay flag is set. Even if this is the case, underflow can be a problem if it was not the encoder's intention to cause an underflow, since it may cause future decoder buffer overflow. For this reason, we assume in this section that underflow is forbidden.

Let $B_i^{d(d)}$ be the actual decoder buffer fullness when the bits corresponding to the i-th frame are removed from the decoder buffer, and $B_i^{d(e)}$ be the nominal decoder buffer fullness as computed by the encoder, assuming no jitter. Let R_{i+L} be the number of bits which would arrive at the buffer during the i-th interval, assuming no jitter. E_i is the number of bits that leave the buffer during the i-th interval, which is also the number of bits used to encode frame i.

In general, the encoder computes $B_i^{d(e)} = B_{i-1}^{d(e)} + R_{i+L} - E_i$. However, if data arrive too late, the buffer may be less full than the encoder expected, in which case, $J_{max} R_{i+L}$ less data will have been received by the decoder buffer. So the actual decoder buffer fullness, $B_i^{d(d)}$, will

differ from that expected by the encoder, $B_i^{d(e)}$:

$$B_i^{d(d)} \geq B_i^{d(e)} - J_{max} R_{i+L}. \tag{1}$$

Similarly, if data arrives too early, the decoder buffer may be fuller than expected. In this case, the fill-rate of the buffer is now R_{i+L+1}.

$$B_i^{d(d)} \leq B_i^{d(e)} + J_{max} R_{i+L+1}. \tag{2}$$

Equations (1) and (2) can be combined to obtain bounds on the actual decoder buffer fullness given the decoder buffer fullness computed at the encoder and the actual maximum jitter.

If the encoder and multiplexer know the correct value of J_{max}, they can ensure that the decoder's buffers will not underflow or overflow by satisfying the following equation:

$$J_{max} R_{i+L} \leq B_i^{d(e)} \leq B_{max}^d - J_{max} R_{i+L+1}. \tag{3}$$

where B_{max}^d is the physically available buffer space.

If the future transmission rate R_{i+1} is unknown at the encoding time i, an upper bound can be used in its place. The more accurately R_{i+1} is known, the more fluctuations can be allowed in the encoded rate and hence better video quality can be produced. One way to view the above equation is to think of the encoder as keeping the VBV from getting too full or too empty.

In a real system, the encoder may not know the correct value of J_{max}. The decoder must compute and send this information to the encoder. It is possible that the decoder may not observe (and hence be able to transmit) the maximum jitter to the encoder until after it occurs. The decoder may also not detect that jitter is occurring in the unlikely event that all packets with Enc_CRs are jittered identically. Furthermore, there is some transmission delay between the decoder and the encoder. Therefore, decoder buffer overflow or underflow could occur before the encoder receives the correct value of J_{max}.

One solution is to have the decoder start with a large estimate of jitter and reduce it over time. In addition, if the encoder *does* adjust its coding parameters based upon available jitter information, the probability of overflow or underflow is minimized. Furthermore, if the decoder can estimate the maximum jitter exactly, the encoder can restrict the interval during which overflow and decoder problems can occur to the transmission delay interval.

The preceeding analysis assumed that the decoder clock was stable and accurate. However, even if the decoder clock is stable with the correct frequency, it could have a constant offset of as much as J_{max}. For example, in the pathological case that all packets containing encoder clock references, Enc_CR, are all delayed by J_{max}, the decoder's time clock, D_TC, would be slow by J_{max}. If the clock is slow, the decoder would remove data from its buffer later than expected by the encoder, and if the clock is slow, data would be removed too soon. If this is taken into account, Eq. (3) becomes

$$2J_{max} R_{i+L} \leq B_i^{d(e)} \leq B_{max} - 2J_{max} R_{i+L+1}. \tag{4}$$

These differences could be accommodated by using a overestimate of the jitter value in the encoder.

To prevent encoder buffer overflow and underflow [2],

$$0 \leq B^e_{i-1} + E_i - R_i \leq B^e_{max} \tag{5}$$

$$R_i - B^e_{i-1} \leq E_i \leq R_i + B^e_{max} - B^e_{i-1} \tag{6}$$

which is a constraint on the number of bits per coded frame for a given channel rate. Alternatively, we can instead let the number of bits per frame E_i be unconstrained, and force the channel rate R_i to accommodate [2]:

$$0 \geq -B^e_{i-1} - E_i + R_i \geq -B^e_{max}$$

$$E_i - (B^e_{max} - B^e_{i-1}) \leq R_i \leq B^e_{i-1} + E_i$$

To prevent decoder buffer overflow and underflow we now have

$$J_{max}R_{i+L} \leq B^{d(e)}_{i-1} + R_{i+L} - E_i \leq B^{d(e)}_{max} - J_{max}R_{i+L+1} \tag{7}$$

$$R_{i+L} + B^{d(e)}_{i-1} - B^d_{max} + J_{max}R_{i+L+1} \leq E_i \leq R_{i+L} + B^{d(e)}_{i-1} - J_{max}R_{i+L} \tag{8}$$

which is a constraint on the encoder bit-rate for a given channel rate. Alternatively, we can again allow the number of bits per frame to be unconstrained, and examine the constraint on the channel rate R_i.

$$E_i - B^{d(e)}_{i-1} \leq R_{i+L}(1 - J_{max}) \leq E_i + \left(B^d_{max} - B^{d(e)}_{i-1}\right) - J_{max}R_{i+L+1}$$

or, for $i > L$

$$\underbrace{E_{i-L} - B^{d(e)}_{i-L-1} \leq R_i(1 - J_{max})}_{\text{decoder underflow condition}} \leq \underbrace{E_{i-L} + \left(B^d_{max} - B^{d(e)}_{i-L-1}\right) - J_{max}R_{i+L+1}}_{\text{decoder overflow condition}} \tag{9}$$

The upper bound on R_i on the right side of Eq. (9) depends on R_{i+1}, which may not be known before estimating R_i. However, an upper bound on R_{i+1} can be obtained from a channel constraint like the leaky bucket. It can also be used to obtain an additional constraint on R_i:

$$R_{i+1}J_{max} \leq E_{i-L} + \left(B^d_{max} - B^{d(e)}_{i-L-1}\right) - (1 - J_{max})R_i$$

or

$$R_iJ_{max} \leq \left[E_{i-L-1} + \left(B^d_{max} - B^{d(e)}_{i-L-2}\right) - (1 - J_{max})R_{i-1}\right]/J_{max} \tag{10}$$

With these constraints and the techniques in [2], selection of the encoded bit rates E_i and transmitted bit rates R_i can be made to ensure no encoder or decoder buffer overflow or underflow, when the jitter is less than J_{max}.

2.2 Transport buffer

Underflow is allowed for the transport buffer, and since the exit-rate from the transport buffer is at least 18 Mbps (for Main Profile), overflow is a concern only if the transmission rate may be close to 18 Mbps.

For the multiplexing buffer, there are two ways to describe the exit-rate: the leak method and the vbv_delay method [4]. If the leak method is used, underflow is allowed. However, the exit rate cannot be precisely known at the encoder/multiplexer if there is jitter, since data transfers from the multiplexing buffer to the elementary stream decoder buffer only if the latter is not full. If the vbv_delay method is used, neither underflow nor overflow is allowed.

3 Decoder

We propose that the decoder optionally send four pieces of information to the encoder: an estimate of the received jitter, the available physical buffer size (once), information about frames skipped due to decoder buffer underflow, and finally information about network congestion received in the ATM cells.

The decoder must estimate the maximum received jitter so it can send this information to the encoder. One possible estimate is to set $J_{dec} = \max\{J_{dec}, |SCR - STC|\}$. However, the resulting estimate of the jitter, J_{dec}, may be smaller than the actual maximum jitter, J_{max} if no packets containing an Enc_CR have been received yet with the maximum jitter, or if the decoder clock is either somewhat fast or somewhat slow. Also, the estimate J_{dec} may overestimate J_{max} by a factor of 2 if the decoder clock is fast by J_{max} and a received packet is delayed by J_{max}, or if the decoder clock is slow and a received packet is early.

More sophisticated methods to estimate the maximum received jitter at the decoder could also be envisioned. In particular, it may be possible that the maximum jitter could be estimated iteratively, as shown in Fig. 1.

The decoder also sends information to the encoder about the frames that have been skippped because of decoder buffer underflow. Some frames are skipped intentionally at the encoder, while other frame skips may be unintentionally caused by jitter. The encoder must become aware of any frames skipped in the latter case to ensure no subsequent decoder buffer overflow occurs.

If a frame is intentionally skipped, a conforming encoder would use the correct time stamp for the subsequent frame. Therefore, the decoder can detect when a frame is unintentionally skipped by examining the received decoding and presentation time stamps.

4 Encoder

The encoder receives the decoder's estimate of the maximum delay jitter, J_{dec}. It then has the option to adjust its coding parameters to minimize the probability of future buffer overflows or underflows.

Jitter can affect the buffer fullness of all three buffers in the RTD model: the transport buffer, the multiplexing buffer, and the elementary stream decoder buffer. In each case, the general principles are as described in Section 2.1 above. Overflowing the transport buffer is a

concern only if the transmission rate is on the order of $1.2 \times R_{max}$, where R_{max} is the maximum rate allowed by the profile. Preventing the multiplexing buffer from overflowing or underflowing is straightforward using the Eq. (3).

To keep the elementary stream decoder buffer from overflowing or underflowing, the encoder has two options: it can either adjust the encoding and transmission rates such that equation (4) is satisfied (using J_{dec} as an estimate of J_{max}), or adjust the buffering delay as described below.

4.1 Elementary stream decoder buffer

The decoder sends the encoder the information described in the previous section, including an estimate of the received maximum jitter, J_{dec}, and the available physical decoder buffer size for smoothing and jitter, EB_{max}. Information about decoder frame-skipping is also sent. The encoder knows the amount of buffering delay in the codec, LT, since it controls this when coding the Decoding Time Stamps (DTS) and Presentation Time Stamps (PTS).

The encoder uses the received value of J_{dec} to determine how it will adjust its parameters. If $J_{dec} << LT$, then the encoder uses Eq. (4) with J_{max} replaced by J_{dec}, along with the techniques presented in [2] to adjust its coding parameters to prevent decoder buffer underflow or overflow even in the presence of jitter. Otherwise, the encoder has the option of increasing the delay in the system to reduce the probability of decoder buffer underflow.

4.1.1 Encoder-induced buffering delay variations

If $J_{dec} \approx LT$, then decoder buffer underflow is probable. However, the encoder can increase the delay by forcing the decoder to skip a picture (provided the parameter low_delay=1). If the encoder does not intentionally increase the delay, the decoder may do so automatically when decoder buffer underflow occurs. However, the encoder must be aware of any frame skipping at the decoder to prevent future decoder buffer overflow.

We describe how the encoder can enforce buffering delay variations using two timelines. The top time line shows when bits corresponding to frame n start to enter the encoder buffer. The bottom time line shows when bits from frame n exit the decoder buffer.

Suppose the buffering delay is 1 frame, and the encoder wants to increase it to 2 frames at frame n. That is,

```
n-2  n-1  n    n+1  n+2  n+3
     n-2  n-1  -    n    n+1
```

Then the encoder can select R_n to be too small to get all of the encoded bits from frame n, E_n, to the decoder in time, even if all packets are early by J_{dec}. Alternatively, the encoder could force decoder buffer underflow by selecting E_n sufficiently high that it can't be received at the decoder buffer with the available rate, R_n. The encoder then continues to select E_i and R_i based on equations (xx) and (4) using the new longer delay parameter L.

Now suppose at some later time the encoder wants to decrease the buffering delay, perhaps because the observed jitter has decreased.

```
n-2  n-1  -    n+1  n+2  n+3
          n-2  n-1  n+1  n+2
```

Then, it can not code one picture and resume coding at the next. In each case, the encoder also puts on the correct timestamps (DTS and PTS).

In addition to the above, the encoder also controls the fullness of the decoder buffer using well-known techniques [2] to ensure that no overflow occurs. Even if the transmission rate is constant, if the delay varies as above, the encoder must monitor both the encoder buffer fullness and the decoder buffer fullness and adjust the encoded bit-rate and transmitted bit-rate accordingly.

Sometimes, frame skipping can occur at the decoder against the encoder's intention, whether because of a sudden increase in the jitter, an inaccurate estimate of jitter by the decoder, or because the correct jitter value is in transit from the decoder to the encoder. If this happens, once the encoder has become aware of the frame-skipping, the encoder has the option of either continuing to encode with the newly increased buffering delay, or shortening the delay back to the original value as described above.

5 Syntax

The decoder sends periodically an estimated maximum jitter using 8 bits, 4 for the exponent E, 4 for the mantissa M. The possible range is 50 usec thru 1000 msec. $J_{dec} = M \times 10^E$.

The decoder also sends, once, available buffer space. Includes the decoder buffer space and transport buffer space. The syntax could be the same as in [4].

Decoder periodically sends information about skipped frames caused by decoder buffer underflow. This allows the encoder to know when the decoder may have unavoidably increased the buffering delay, and is necessary so the encoder can prevent future decoder buffer overflow.

6 Extensions

If the compressed video is stored, it is still possible to reduce the bit-rate using techniques presented by BT Labs and Motorola at the recent Packet Video Workshop (papers D17 and D18). The benefit is that while the video quality will be degraded more than if the encoding parameters were modified by a real-time encoder, it will be degraded less than if buffer overflow or underflow occurred because of excess jitter.

References

[1] "ISO 13818 Working Draft of ISO-IEC/JTC1/SC29/WG11 Moving Picture Experts Group, part 9", Real-Time Interface Specification.

[2] A. R. Reibman and B. G. Haskell, "Constraints on Variable Bit-Rate Video for ATM Networks," *IEEE Transactions on Circuits and Systems for Video Technology*, volume 2, number 4, pages 361–372, December 1992.

[3] B. G. Haskell and A. R. Reibman, "Timing Recovery for Variable Bit-Rate Video on ATM Networks", MPEG92/396, AVC-315, Paris France, July 3, 1992.

[4] "ISO 13818 International Standard of ISO-IEC/JTC1/SC29/WG11 Moving Picture Experts Group, part 1", Systems.

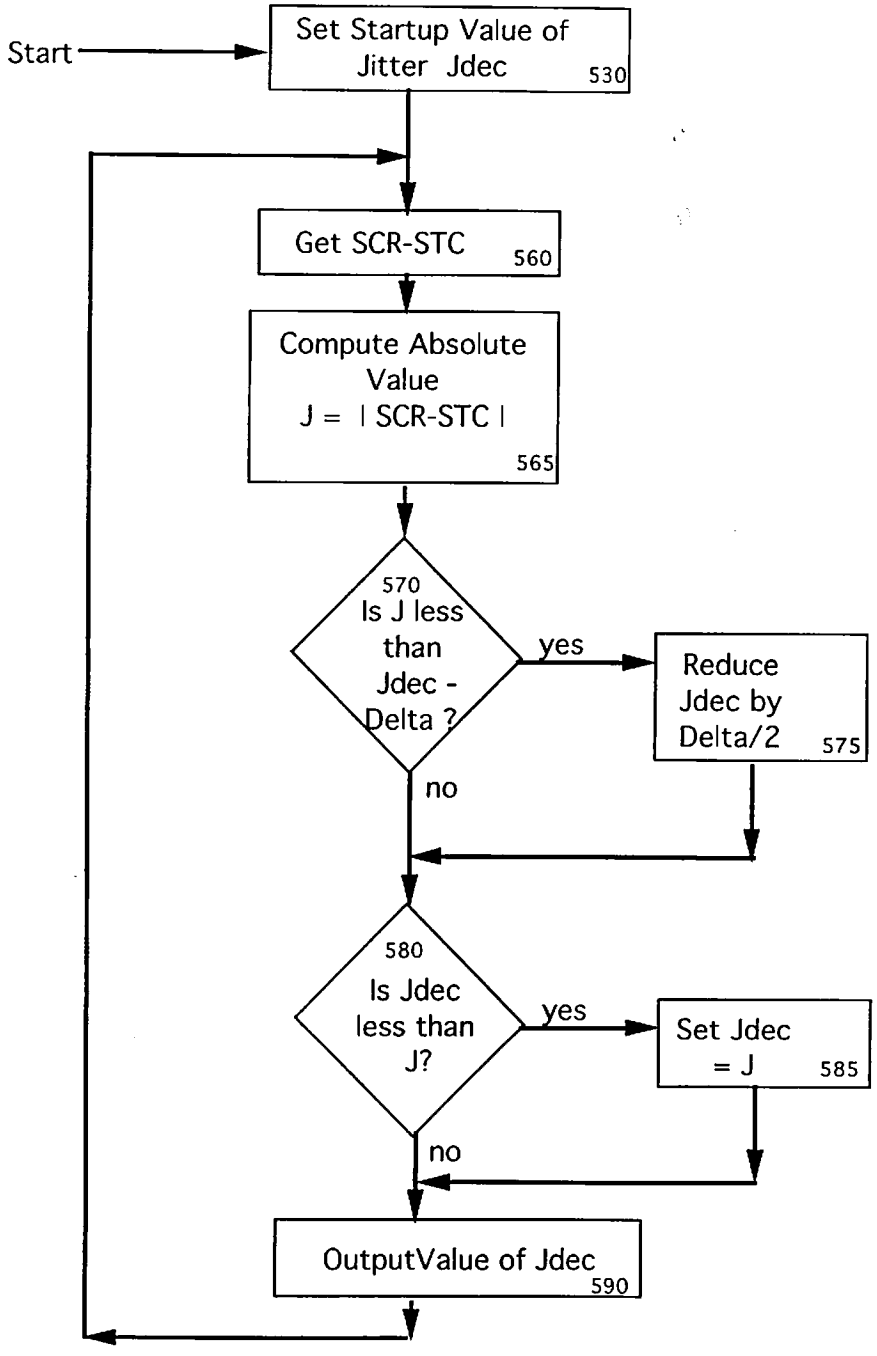

Fig. 1 Flow Chart of Jitter Calculation