Playout management of interactive video - an adaptive approach

Sanjay K. Jha, Peter A. Wright and Michael Fry
Faculty of Maths & Computing Sciences
University of Technology, Sydney, NSW 2007 Australia
phone: + 612 9514 1858, fax: + 612 9514 1807
email: sanjay,mike@socs.uts.edu.au, peterw@maths.uts.edu.au

Abstract

This paper examines problems associated with display of live continuous media. Under the assumption that the network cannot guarantee the required bounds on delay and jitter and the operating system scheduling is non-realtime, there is a need to accommodate the delay and jitter in the end systems in order to maintain a desirable Quality of Service (QoS). We propose a method of video playback which requires accurate estimation of display cycle time of video frames and the delay suffered by frames in the packet network. We apply various deterministic forecasting methods used in time series analysis on experimental data collected from video transmission. Suitable methods are recommended for display cycle time and delay estimation.

Keywords

Multimedia, playout, QoS, packetised video, time-series

1 INTRODUCTION

There has been a dramatic increase in the processing power of workstations and the bandwidth of high speed networks. This has given rise to new real-time applications such as multimedia applications. These applications have traffic characteristics and Quality of Service (QoS) requirements which are quite different from existing data-oriented applications. Delay jitter is an important performance metric for real-time traffic (Difference in end-end-delay is called jitter). Some factors responsible for causing jitter are:

- the time required to digitise, compress and decompress video frames;
- variation in delay in the network transport system;
- operating system scheduling latencies.

Under the assumption that the network cannot guarantee the required bounds on delay and the operating system scheduling latency exists, there is a need to accommodate the delay jitter in the end systems. If the underlying network is connection less (individual packets may take different routes), the video frames may arrive at the destination out of sequence or after the time at which they should be displayed. A gap may result if a frame is not available for display. This may affect the quality of audio/video playout. In our experiments we consider that a frame that is to be displayed at time t , must be completely present in the buffer at that time. It is then removed instantaneously, decoded and rendered (we will call the decoding, decompress and render time of video frame as display cycle). In practice, it would be displayed a short time later, depending on the speed of the decoding process.

The rest of this paper is organised as follows. Section 2 describes an existing playout algorithm and its adaptation for video. Section 3 presents smoothing methods used to estimate the display cycle time. Section 4 presents results of applying these smoothing methods to traces collected for JPEG and MPEG video. Section 5 demonstrates the suitability of a new method for network delay estimation and spike detection.

2 ADAPTIVE PLAYOUT ALGORITHM

In this section we describe an adaptive method of video playout which has been adapted from audio playout method [Ramjee *et al.*, 1994].

The playout time for first packet P_i is calculated using :

$$P_i = S_i + DA + n * j. \quad (1)$$

where

P_i = Playout time for first frame i

S_i = Send time for first frame i

$n = 1, 2, \dots$

j = jitter

DA = Adapted delay value

The playout point of all subsequent packets is calculated using :

$$P_k = P_i + S_k - S_i. \quad (2)$$

where $k = 2, 3, \dots$

For audio packets, playout time and display time are assumed to be the same as the time to playout the audio packet is very small.

Upon receipt of first video frame (in contrast to a packet in audio), playout time is estimated using the following formula:

$$P_i = S_i + DA + n * j - DR_i. \quad (3)$$

where, DR_i = Average of display cycle

In this approach we store the compressed video frames in a receiver buffer as they arrive [Jha and Fry, 1996]. A delay factor $DA + n * j$ is added to the send time of frame. Section 5 describes the method of calculating the delay DA . The playout time for subsequent frames can be calculated in two ways. In the case of non-interactive sessions such as video on demand applications (where $DA + n * j$ can be as large as one second) subsequent frames are calculated as

$$P_k = S_k + DA + n * j + IG - DR_k. \quad (4)$$

where

$$k = 2, 3 \dots$$

IG = Interframe gap (If the frame rate is 25 fps then $IG = 1000/25 = 40(ms)$).

For interactive sessions such as video-conferencing where there is a maximum bound on end to end latency and IG is not know in advance, it would be preferable to maintain the timing relationship of arriving frames. A modified equation(2) is used to calculate the playout time of subsequent frames

$$P_k = P_i + S_k - S_i - DR_k. \quad (5)$$

Figure 1 Video playout method.

In Figure 1 the playout point is DR units before the actual display time (dark black rectangles on the Display line of figure). If the estimate of DR

is accurate, we should be able to display frames close to the display time. In the case of continuous voice traffic, the factor $DA + n * j$ is updated only between talkspurts. Video or music do not have talkspurts and silence periods. Campbell et al. [Campbell and Coulson, 1996] suggest changing this factor at the beginning of each group of pictures (GOP) for video stream using MPEG compression.

In the next section, we present some well known deterministic forecasting methods used in time series analysis. We apply these methods on experimental data collected for video display cycle time in order to select the best method of forecasting.

3 FORECASTING METHODS

In this section we present three smoothing methods used to estimate the display cycle. These deterministic methods have been selected because of their suitability to forecasting large numbers of items. The computing resources used by these smoothing methods are low in terms of both CPU and memory. Basu et al. [Basu *et al.*, 1996] have used Auto-Regressive-Moving-Average (ARMA) to model the internet traffic. This model is not suitable for real-time scenario where updates are frequent.

(a) Single Exponential Smoothing

Single Exponential Smoothing (SES) is an attractive method of forecasting as it is computationally simple. The following equation is used to calculate the forecasted value:

$$F_{t+1} = \alpha X_t + (1 - \alpha)F_t, \quad 0 \leq \alpha \leq 1. \quad (6)$$

where,

F_t = Forecasted value for t th period

X_t = Current value

α = Parameter chosen by user

(b) Adaptive Response-Rate Single Smoothing

The Adaptive-Response-Rate Single Exponential Smoothing Method (ARRSES) allows the value of α to change in response to changes in data pattern. The α value is automatically changed from period to period to allow for changes in the structure of data. Fluctuations in α can be controlled by putting an upper bound on how much α is allowed to change from one period to the next [Makridakis *et al.*, 1983].

F_{t+1} is calculated using equation(6) where,

$$\alpha_{t+1} = \left| \frac{E_t}{M_t} \right|. \quad (7)$$

$$E_t = \beta e_t + (1 - \beta)E_{t-1}, \quad 0 \leq \beta \leq 1. \quad (8)$$

$$M_t = \beta |e_t| + (1 - \beta)M_{t-1}. \quad (9)$$

where

β = Parameter chosen by user
and e_t is the error term defined by equation

$$e_t = X_t - F_t. \quad (10)$$

(c) Holt's Two-Parameter Method

Two-parameter forecasting was done with Holt's linear exponential smoothing, which smoothes the trend values separately. This provides greater flexibility, since it allows the trend to be smoothed with a different parameter than that used on the original series. The forecast can be found using two smoothing constants (with values between 0 and 1) and the three equations:

$$S_t = \alpha X_t + (1 - \alpha)(S_{t-1} + b_{t-1}), \quad 0 \leq \alpha \leq 1. \quad (11)$$

$$b_t = \gamma(S_t - S_{t-1}) + (1 - \gamma)b_{t-1}, \quad 0 \leq \gamma \leq 1. \quad (12)$$

$$F_t = S_t + b_t. \quad (13)$$

where

S_t = Smoothed Value for t th period

α, γ = Parameters chosen by user

4 DISPLAY CYCLE TIME ESTIMATE

4.1 Analysis of JPEG Data

In order to evaluate the best approach for forecasting these values, we measured the display cycle time for two different video clips using JPEG compression on Sun Ultra Sparc-1 machines. The *lion.jpg* clip is a stampede fragment

Figure 2 Frame Size vs Display Cycle Time.

(5925 frames) from the movie *Lion King* with lots of movement and frequent scene changes. The second clip *song.jpg* is from a popular English song which has comparatively less movement and scene changes (2558 frames). *Vic* was used to decompress and render the frames [McCanne and Jacobson, 1995]. Frames were received from another workstation on the same local area network at an average of 25fps rate. Frame size in both cases is 352×288 pixels (CIF).

(a) Frame Size and Display cycle Time

In order to estimate the display cycle we plotted frame size (compressed) versus the display cycle time of the two video clips. Figure 2 shows these plots. As is evident from the plots, there is little correlation between the frame size and display cycle time.

(b) Model Comparison Criterion

We will compare our models on the basis of their Mean Absolute Deviation (MAD). This is the arithmetic mean of the absolute deviations of each forecast from its actual value. That is,

$$MAD = \frac{\sum_{t=1}^n |X_t - F_t|}{n}. \quad (14)$$

Cen et al [Cen *et al.*, 1995] used Mean Squared Error (MSE) to compare the effectiveness of a particular playback method. MSE squares the errors and so penalises a forecasting technique much more for large errors than for small errors. On the other hand, MAD penalises all errors in direct proportion to their absolute size. In the context of interactive video playback, we find little a priori justification for using squared error penalties. On a purely practical level, the MSE is also very susceptible to influence by "outliers", unlike the MAD. For these reasons we used only the MAD criterion in our experiments.

Figure 3 Frame Sequence No. vs Display Cycle Time for 1st 500 frames.

Table 1 Mean absolute deviation for JPEG models.

Method	Parameters	Mean Absolute deviation	
		<i>lion.jpg</i>	<i>song.jpg</i>
SES	$\alpha = .125$	2.45	3.96
	$\alpha = .4$	2.16	3.36
	$\alpha = .6$	2.26	3.30
	$\alpha = .8$	2.46	3.33
HOLT	$\alpha = .4 \gamma = .1$	2.22	3.53
	$\alpha = .6 \gamma = .1$	2.34	3.48
	$\alpha = .8 \gamma = .2$	2.66	3.66
ARRSES	$\beta = .2$	2.09	3.49
	$\beta = .2 \text{ limit} = .6$	2.11	3.37

4.2 Models Tested

Most commonly researchers have used the digital filters which are SES forecasts with $\alpha = 1/16$ or $\alpha = 1/8$. We calculated the forecasted values using these parameters as well as $\alpha = .4, .6, .8$. Since the display cycle time had an apparent trend (Figure 3), we also tried Holt's two-parameter forecasting method. A wide variety of (α, γ) pairs were tried with the best choice being $(.4, .1)$.

Table 1 shows the calculated Mean Absolute Deviations using the various forecasting methods. The ARRSES with $\beta = .2$ performs best for *lion.jpg* ($MAD = 2.09$). The SES ($\alpha = .4$) is very close ($MAD = 2.11$). For *song.jpg* SES($\alpha = .6$) gives the lowest MAD (3.30). Plots of the forecast errors did not show any noticeable difference between the various forecasting methods [Jha and Wright, 1997].

4.3 Analysis of MPEG Frames

In this section we analysed traces collected for 900 frames from an MPEG clip *ftt.mpg* (240×352 pixels). The *ftt.mpg* movie was coded using the following sequence IBBPBBPBBPBBPBB. Since *Vic* does not support MPEG format, we used the Berkeley *Mpeg-play* tool to collect traces [Rowe and Smith, 1992]. *Mpeg-play* doesn't support network communication and hence the display cycle time was not affected by packet arrival.

(a) MPEG Frame Size and Display Cycle Time

In order to estimate the display cycle we plotted frame size (compressed) versus the display cycle time of the video clip *ftt.mpg*. Figure 4 shows that the frame size and display cycle time seem to be correlated. We did not examine this phenomenon any further as our results for JPEG frames in the networking environment (using *Vic*) were not highly correlated. Also any correlation between the frame size and display cycle time will be affected by the processing power of a host system.

Figure 4 Frame Size(bytes) vs Display Cycle time.

Figure 5 Display Cycle Time for I, P and B Frames *ftt.mpg*.

(b) Models Tested

The MPEG display cycle times were forecast using the same models as for the JPEG clips. The initial results are shown in the Table 2. Note that the MPEG stream consists of I, P and B frames and decoding time for these 3 types of frames vary. The averages for display cycle time of I, P and B frames are 36.7, 31.1 and 22.6 ms for the *ftt.mpg* clip. We plot the decode time of these frames separately in Figure 5. The figure clearly shows that I frame takes longest followed by P and then B. We then used the smoothers separately on each type of frame. From the results in Table 3 it is evident that the value of MAD has dropped substantially for every type of frame. We conclude that forecasts of the 3 types of frames (I, P & B) should be stored separately as this provides a much better estimate. Table 3 shows that SES

Table 2 MAD for MPEG's IPB frames (combined).

<i>Method</i>	<i>Parameters</i>	<i>MAD(flt.mpg)</i>
SES	$\alpha = .125$	4.68
	$\alpha = .4$	5.34
	$\alpha = .8$	6.43
HOLT	$\alpha = .4 \ \gamma = .1$	5.65
	$\alpha = .6 \ \gamma = .1$	6.14
	$\alpha = .4 \ \gamma = .4$	5.78
	$\alpha = .2 \ \gamma = .8$	5.31
ARRSES	$\beta = .2$	5.97

($\alpha = .4$) performs better for P and B frames (in terms of MAD) in comparison to ARRSES but is marginally worst for I frames.

Table 3 MAD for MPEG I,P and B frames.

<i>Clip flt.mpg</i>	<i>Method</i>	<i>Frame count</i>	<i>Parameters</i>	<i>MAD</i>
<i>I - frame</i>	SES	69	$\alpha = .4$	2.67
	ARRSES		$\beta = .2$	2.57
<i>P - frame</i>	SES	231	$\alpha = .4$	1.15
	ARRSES		$\beta = .2$	1.2
<i>B - frame</i>	SES	600	$\alpha = .4$	1.06
	ARRSES		$\beta = .2$	1.08
<i>I - frame</i>	SES	69	$\alpha = .4$	2.67

5 DELAY ESTIMATION/SPIKE DETECTION ALGORITHM

In the section 3 we looked at an adaptive method of playout for audio-video frames. The success of this method depends on the accuracy of estimating/forecasting the value of the end to end delay of packets. Analysis of delay characteristics has been carried out by several researchers [Bolot, 1993] by collecting traces from the Internet. These traces show occurrence of frequent spikes which last for several milliseconds. A spike is characterised by a sudden large increase in the end to end delay of packets. The delay trace plotted in Figure 6 was collected between a host at the University of Technology, Sydney (mill.socs.uts.edu.au) and Columbia University, New York (sutton.cs.columbia.edu). These traces were collected using a modified version of the *ping* utility which sends a packet to specified destination every 20 ms. The number of hops followed by each packet was more than 20. The

Table 4 Mean absolute deviations for delay.

<i>Method</i>	<i>Parameters</i>	<i>Mean Absolute Deviation</i>	
		<i>normal</i>	<i>spike</i>
SES	$\alpha = .125$ (RFC 793)	5.70	25.24
	$\alpha = .4$	3.05	10.30
HOLT	$\alpha = .4 \ \gamma = .1$	3.62	11.94
ARRSES	$\beta = .2$ no limit	3.24	8.21

spike was created artificially to resemble the spikes observed by Ramjee et al [Ramjee *et al.*, 1994].

5.1 Conventional Delay Estimation/Spike Detection Algorithm

Most audio tools estimate the delay factor using estimators described by Jacobson [Jacobson, 1988]. The equations used for this purpose are similar to equations (6),(8),(10).

An adaptive algorithm which explicitly adjusts to sharp spikes like increases in packet delay has been suggested by Ramjee. If there is no spike then delay is updated using equation 6. In this method a threshold value of delay is used to detect the spike. The threshold value is arbitrarily set to 800ms. If a spike is detected, a different formula is used to estimate delay :

$$F_{t+1} = F_t + X_{t+1} - X_t. \quad (15)$$

5.2 Proposed Delay Estimation/Spike Detection Algorithm

We propose the self-adjusting ARRSES method, described in section 3, which estimates delay value and adapts to spikes automatically. There is no need to set any threshold value nor provide a separate formula to deal with spikes. Table 4 shows the Mean Absolute Deviation (MAD) for the trace without spikes (normal) and with spike. In terms of MAD, the SES method with $\alpha = .4$ performs better than the other methods but is only marginally better than ARRSES. For the trace with spike, ARRSES outperforms all the other methods. Figure 6 shows the comparison of ARRSES method with SES. SES method with $\alpha = .125$ performs worst. Holt's 2 parameter over estimates the values most of the time.

In order to verify the suitability of the ARRSES method for spikes of lesser magnitude, delays were forecasted for a simulated trace containing spikes of different magnitudes. These spikes ranged down from the spike in Figure 6

Figure 6 ARRSSES .2 vs SES .4

which was about four times the average delay, to a spike which was only about twice the average delay. The ARRSSES method tracks spikes of any magnitude with success [Jha and Wright, 1997].

6 CONCLUSION AND FUTURE WORK

This paper has considered problems associated with the playback of video streams. In particular the paper has considered display cycle time estimation as important factor in achieving a required quality of service. Estimation of display cycle time of video frames was performed using several forecasting methods. We conclude that the SES with $\alpha = .4$ to $.6$ works well for video frames. We have also demonstrated that the ARRSSES works well for estimating end-to-end delay of packets even in the presence of small and large spikes. One possible disadvantage in using the ARRSSES is that the α value can fluctuate a lot. This can be overcome by putting an upper bound on how much α can change from one period to the next. Future work will include evaluation of bounds on α .

ACKNOWLEDGEMENT

Authors would like to thank Atanu Ghosh from University of Technology, Sydney (UTS) and Varuni Witana from Distributed Systems Technology Centre, UTS, for valuable discussions.

REFERENCES

- [Basu *et al.*, 1996] Sabyasachi Basu, Amarnath Mukherjee, and Steve Klivansky. Time series models for internet traffic. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, San Fransisco, California, March 1996.
- [Bolot, 1993] Jean-Chrysostome Bolot. Characterizing end-to-end packet delay and loss in the Internet. *Journal of High Speed Networks*, 2(3):305–323, 1993.
- [Campbell and Coulson, 1996] Andrew Campbell and Geoff Coulson. Implementation and evaluation of the qos—a transport system. In Dabbous Walid and Diot Christophe, editors, *Protocols for High-Speed Networks V*, pages 201–218, Inria, France, October 1996. Chapman and Hall.
- [Cen *et al.*, 1995] S. Cen, C. Pu, R. Staehli, C. Cowan, and J. Walpole. A distributed real-time MPEG video audio player. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Lecture Notes in Computer Science, pages 151–162, Durham, New Hampshire, April 1995. Springer.
- [Jacobson, 1988] Van Jacobson. Congestion avoidance and control. *ACM Computer Communication Review*, 18(4):314–329, August 1988. Proceedings of the Sigcomm '88 Symposium in Stanford, CA, August, 1988.
- [Jha and Fry, 1996] S. K. Jha and M. Fry. Continuous media playback and jitter control. In *Proc. of the IEEE Conference on Multimedia Systems*, Hiroshima, Japan, June 1996.
- [Jha and Wright, 1997] S. K. Jha and P. A. Wright. Adaptive playout management of interactive video streams. Technical report, School of Computing Sciences, University of Technology, Sydney, Australia, March 1997.
- [Makridakis *et al.*, 1983] S. Makridakis, S. C. Wheelwright, and V. C. McGee. *Forecasting: Methods and Applications*. John Wiley and Sons, 2 edition, 1983.
- [McCanne and Jacobson, 1995] Steve McCanne and Van Jacobson. vic: A flexible framework for packet video. In *Proc. of ACM Multimedia '95*, November 1995.
- [Ramjee *et al.*, 1994] Ramachandran Ramjee, Jim Kurose, Don Towsley, and Henning Schulzrinne. Adaptive playout mechanisms for packetized audio applications in wide-area networks. In *Proceedings of the Conference on Computer Communications (IEEE Infocom)*, pages 680–688, Toronto, Canada, June 1994. IEEE Computer Society Press, Los Alamitos, California.
- [Rowe and Smith, 1992] Lawrence A. Rowe and Brian C. Smith. A continuous media player. In *Third International Workshop on network and operating system support for digital audio and video*, pages 334–344, San Diego, California, November 1992. IEEE Computer and Communications Societies.