Video TFRC

Evan Tan, Jing Chen, Sebastien Ardon, Emmanuel Lochin

Video TFRC

Evan Tan[+], Jing Chen[*], Sebastien Ardon[*], and Emmanuel Lochin[#]
[+]University of New South Wales, Sydney, Australia
[*]NICTA, Sydney, Australia
[#]Université de Toulouse, DMIA, ISAE, France
{evan.tan, jing.chen, sebastien.ardon}@nicta.com.au, emmanuel.lochin@isae.fr

***Abstract*— TCP-friendly rate control (TFRC) is a congestion control technique that trade-offs responsiveness to the network conditions for a smoother throughput variation. We take advantage of this trade-off by calculating the *rate gap* between the theoretical TCP throughput and the smoothed TFRC throughput. Any rate gain from this rate gap is then opportunistically used for video coding. We define a frame complexity measure to determine the additional rate to be used from the rate gap and then perform a rate negotiation to determine the target rate for the encoder and the final sending rate. Results show that although this method has a more aggressive sending rate compared to TFRC, it is still TCP-friendly, does not contribute too much to network congestion and achieves a reasonable video quality gain over the conventional method.**

Keywords - complexity measure, congestion control, cross-layer design, H.264/AVC, rate control, TCP-friendly, multimedia streaming

I. INTRODUCTION

With the proliferation of streaming multimedia applications used in the current best-effort Internet, congestion control techniques are crucial in regulating bandwidth greedy multimedia traffic to avoid a congestion collapse of the network. A widely used congestion control technique in the Internet is additive increase, multiplicative decrease (AIMD) that is provided by the transport control protocol (TCP) [4]. However, in most cases, AIMD is not suitable for streaming multimedia applications due to the large throughput variations and the possible delays incurred through retransmissions.

In order to compete fairly with the majority TCP traffic in the Internet, the concept of "TCP-friendly" was created [2] where the generated network traffic has a behavior close enough to that of TCP traffic in similar conditions thus inheriting the congestion control properties of TCP. One such TCP-friendly technique is TCP-friendly Rate Control (TFRC) [1]. TFRC is an equation based congestion control technique for best effort networks that provides a smoother throughput variation over time, making it more suitable for streaming multimedia applications. A number of TFRC-like techniques for streaming multimedia applications have emerged.

One such TFRC variant by Vieron and Guillemot [5] takes into account of the variable video packet sizes and synchronized the receiver feedback to the video frame rate. Kim et al [6] proposed another variant of TFRC that tradeoffs the smoothness of rate variation and network responsiveness by making use of weighted round-trip time (RTT) and retransmission time out (RTO). Shih et al [7] use the TFRC throughput in their rate control method to adjust the video source rate. Zhu et al [8] proposed a joint source rate control and congestion control technique that may temporarily violate the short-term TCP-friendliness to prevent a decoder buffer underflow at the receiver end. But a compensation factor for the calculation of future TFRC throughputs is used to maintain long-term TCP friendliness. Also, their rate calculation technique is driven by a virtual buffer model and does not opportunistically code at a higher rate.

None of these techniques factored in the *video bit rate characteristic* for the calculation of the TCP-friendly transmission rate. The video bit rate tends to vary according to the *complexity* of the frame data, for example an I-frame would be more complex compared to a P-frame as it results in more bits after compression. The same also applies to *scene changes* and *high motion* scenes in a video sequence as they tend to incur a higher prediction error which results in a lower compression efficiency. Thus a typical video bit rate will have occasional 'pulses', a smoothed transmission rate will reduce these 'pulses' and ends up affecting the video quality.

Furthermore, the aforementioned techniques tend to favor a smoothed rate more than a responsive rate, even though the *difference between the two rates could possibly be exploited*. To highlight this fact, we define the *instantaneous transmission rate* to be the upper-bound of the theoretical TCP rate. We then introduce notion of *rate gap* as the difference between the instantaneous transmission rate and a smoothed TFRC rate. Fig. 2 then illustrates a possible rate gap at the time instant *t*. Note that the shaded rate region contained between the two lines is a *TCP-friendly rate region* i.e. any rate that falls within the region is considered a TCP-friendly rate.

In this paper, we propose a joint source rate control and congestion control method named *Video TFRC (VTFRC)* that, depending on the complexity of the frame and the rate gap, *opportunistically* encodes a frame at a higher bit rate and at the same time perform *rate negotiation* with the TFRC protocol to possibly transmit at a higher bit rate in order to meet the increased Quality-of-Service (QoS) requirement. The remainder of the paper is organized as follows: section II details the proposed VTFRC method, section III shows the results of our experiments and section IV concludes this paper.

NICTA is funded by the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

II. THE PROPOSED VTFRC

A. VTFRC architecture

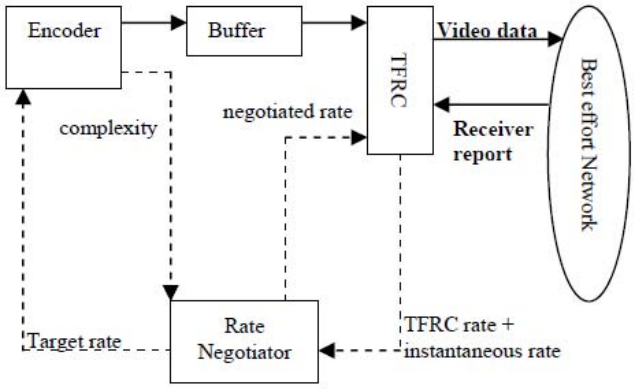

Fig. 1. The VTFRC architecture.

Fig. 1 shows the architecture of VTFRC. Our implementation of TFRC protocol was modified to calculate the instantaneous rate in addition to the standard TFRC rate. The rate negotiator simultaneously receives the complexity of the current frame from the H.264 encoder as well as the current TFRC transmission rate and instantaneous rate from the TFRC protocol. Based on either of these inputs, the rate negotiator will then set a target rate for the H.264 encoder if required by the encoder and return a negotiated rate for the TFRC protocol to transmit at.

B. Rate gap calculation

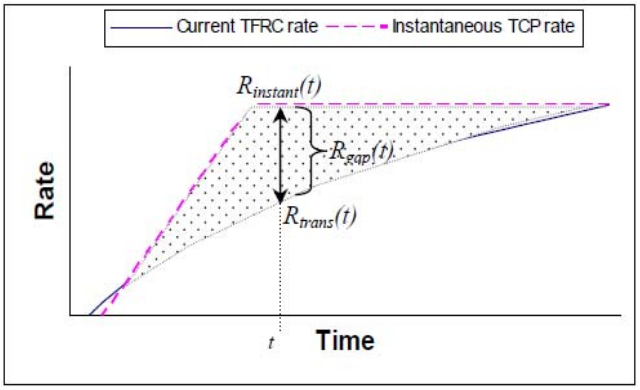

Fig. 2. An illustration of a possible rate gap, the shaded region is the TCP-friendly rate region.

Fig. 2 illustrates how the rate gap is calculated. The TFRC rate ($R_{trans}(t)$) at time t is calculated as per RFC3448 [1] with the *average packet size* used in the calculation as video packets tend to be of variable sizes. To calculate the instantaneous TCP rate ($R_{instant}$), we make use of the theoretical TCP upper-bound [4]:

$$R_{\text{instant}}(t) = \frac{s}{RTT(t)\sqrt{\frac{2p}{3}} + RTO(3\sqrt{\frac{3p}{8}})p(1+32p^2)}, \quad (1)$$

Where s is the average packet size, $RTT(t)$ is the receiver reported RTT as at time t, RTO is the TCP retransmit timeout value and p is the loss event rate. Note that the RTT used for the TCP upper-bound calculation is the current RTT and is not smoothed in order to achieve a slightly more responsive rate.

Since much of the smoothing effect of TFRC comes from the exponentially weighted moving average (EWMA) loss interval calculation for the loss event rate p. We reduce the number of loss intervals used for calculation to 2, so the average loss interval for time t becomes:

$$s_{avg} = \frac{\sum_{i=1}^{2} s_i}{2}, \quad (2)$$

The corresponding loss event rate is then calculated as:

$$p = \frac{1}{s_{avg}}, \quad (3)$$

The loss interval calculation has been simplified to obtain a more responsive effect to recent loss intervals and thus can be used with equations (3) and (1) to obtain the instantaneous rate $R_{instant}(t)$. In short, TFRC is modified by using more recent measurements thus making its rate calculation more reactive to network changes.

Given that at time t, the TFRC rate being transmitted is $R_{trans}(t)$, the rate gap is then:

$$R_{gap}(t) = R_{\text{instant}}(t) - R_{trans}(t). \quad (4)$$

C. Rate negotiation

1) Frame Complexity

Given $R_{gap}(t)$, we make use of the complexity of the current frame, i.e. the possible amount of bits it will take up, to control the amount of additional rate to use as using the whole rate gap might cause more network congestion and larger video quality fluctuations.

Li et al [10] proposed calculating the frame complexity by estimating the mean absolute difference (MAD) of the current frame k based on a linear prediction of the previous frame's (k-1) MAD, where MAD is defined here as the difference between the encoded frame and the original frame. That is:

$$MAD_{pred}(k) = a_1 MAD_{actual}(k-1) + a_2, \quad (5)$$

where a_1 and a_2 are the linear model parameters that are updated via linear regression as described in [9].

The linearly predicted MAD value is then used to calculate the complexity value of frame k by:

$$C(k) = \frac{MAD_{pred}(k)}{\frac{1}{n}\sum_{i=1}^{n} MAD_{actual}(k-i)} - 1, \quad (6)$$

This determines how much more complex the current frame is compared with the past n frames. n is calculated based on the sliding-window data-point selection as described in [9] (the maximum window size is also set to 20 in our implementation). The sliding-window data-point selection will increase the window size when the scene is less complex and decrease the window size when the scene is more complex in order to improve the detection of scene changes within the sequence.

2) Additional TCP Friendly rate

Given that frame k is about to be encoded at time t, the additional rate that can possibly be used for transmission and encoding is:

$$R_{add}(k,t) = C(k) \times R_{gap}(t), \quad (7)$$

Note that $R_{add}(k)$ here is updated at the *start of every frame period* (i.e. just before the frame is being encoded) and thus $R_{gap}(t)$ here is the rate gap at the start of the frame period.

A frame period is used as the updating interval here because the synchronization of the TFRC receiver reports with the encoder frame rate on the test-bed is affected by delay jitter and is difficult to achieve, so we make no assumptions that the TFRC receiver reports will be synchronized.

3) Target rate for encoder

Given that the encoder is currently beginning to encode frame k at time t, the target rate for the encoder rate controller is then calculated by:

$$R_{\text{target}}(k,t) = R_{\text{trans}}(t) + R_{add}(k,t), \quad (8)$$

4) Negotiated rate

Given the current time t and the last frame period for frame k occurred at time t-i the negotiated rate to transmit at is:

$$R_{neg}(t) = \min(R_{trans}(t) + R_{add}(k,t-i), R_{\text{instant}}(t)). \quad (9)$$

III. EXPERIMENTAL RESULTS

A. VTFRC architecture

We implemented our proposed method in the H.264/AVC Joint-Module (JM) 12.2 reference software [14]. The encoder had rate-distortion (RD) optimizations disabled and the rate control mode set to RC_MODE_0, which is the original JM rate control method proposed by Li et al [10]. We set a group-of-pictures (GOP) size of 50 frames with a GOP structure of IPPP and the number of reference frames for motion estimation is set to 1.

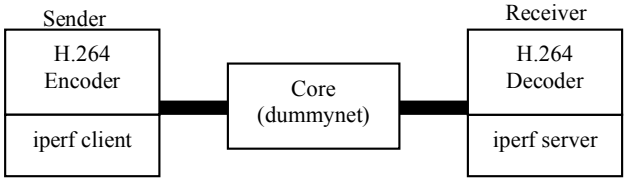

Fig. 3. The test-bed used.

The maximum packet size generated by the encoder is set by limiting the maximum slice size to 1400 bytes. The JM 12.2 reference software puts each slice into one packet and currently does not combine multiple slices to meet a fixed packet size, thus, the packet sizes generated are still highly variable (which is why the average packet size was used in the TFRC rate calculation). At the decoder, we enabled frame copy error concealment. All of the sequences tested (see TABLE I) were of QCIF size (176x144) and a frame rate of 20Hz set for the experiments.

Our experiments were performed on a small test bed with two computers at the endpoints (Fig. 3) emulating the senders and receivers. The middle computer acts as the core of the network with dummynet [11] pipes to emulate the bottleneck bandwidth and RTT. We set the bottleneck bandwidth to be 1.5Mbits/s and the RTT to be 160ms. The buffer overflow of the dummynet queue is set to 100 packets and the simulated loss rate is set to 0%, i.e. any loss generated will be due to queue buffer overflow.

An additional three TCP streams were generated at the sender using iperf [13] with the receiver set as the destination. These three streams therefore compete for buffer space at the core output interface. The TCP streams were given an initial 5 second delay before transmitting, while the encoder transmits right from the start.

TABLE I. NUMBER OF FRAMES FOR EACH SEQUENCE

Sequence	Frames
Container	300
Foreman	400
Grandma	870
Mother-Daughter	961
Salesman	449

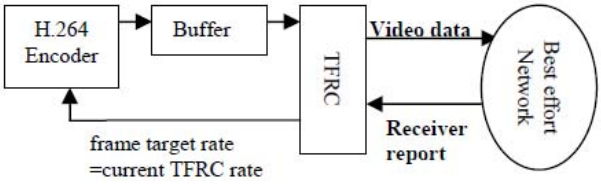

Fig. 4. JM-TFRC sender-end setup.

Our proposed method (VTFRC) is compared to a setup of the original JM rate control method [10] and the standard TFRC protocol which we call JM-TFRC here (see Fig. 4). In JM-TFRC, the rate controller sets the target bit-rate for each frame based solely on the rate given by the TFRC protocol.

B. *TCP-Friendliness Evaluation*

We first check that VTFRC emits TCP-friendly flows. To do this we used the TCP-friendliness ratio (F) metric by Padhye et al [12]. In our case, given the average throughput of the three competing TCP clients, TC_1, TC_2 and TC_3. We calculate the average competing TCP client throughput by,

$$T_C = \frac{1}{3}\sum_{i=1}^{3} TC_i \qquad (10)$$

So given that the average throughput of VTFRC is T_V, the TCP-friendliness ratio is:

$$F = \frac{T_V}{T_C} \qquad (11)$$

Additionally, as was observed by Zhu et al [8], the multimedia flow here involves packets of variable sizes and the fairness of the flow is related to the packet size, we calculate the sending rate in packets per seconds to remove this bias. The results for TCP-friendliness are shown in TABLE II. The sending throughputs over time for Container and Foreman are shown in Fig. 5 and Fig. 6 respectively

TABLE II. TCP-FRIENDLINESS RATIO (F) RESULTS

Sequence	VTFRC *F*	JM-TFRC *F*
Container	0.92	0.80
Foreman	1.07	0.97
Grandma	0.78	0.57
Mother-Daughter	1.02	0.80
Salesman	1.02	0.93

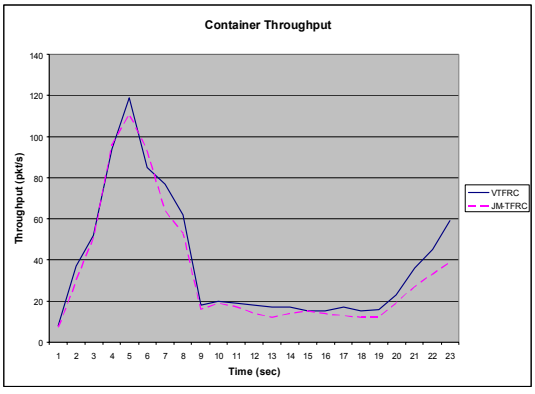

Fig. 5. The sending throughputs over time for Container.

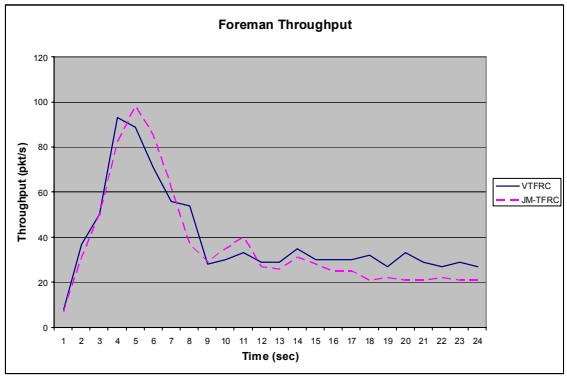

Fig. 6. The sending throughputs over time for Foreman.

It can be seen from the results that VTFRC is a slightly more aggressive transmitter compared to JM-TFRC as all its F values are larger. However, RFC 3448 [1] states that a flow competing with TCP is reasonably fair when its sending rate is within a factor of two of the sending rate of a TCP flow in the same conditions. As the F values of VTFRC are all relatively close to 1, this indicates that the traffic it generates is reasonably fair.

This is expected as VTFRC only makes use of the TCP-friendly rate region to send its additional rate thus it *still produces a TCP-friendly sending rate*.

C. *Video Quality Evaluation*

Next, we investigate the benefit of VTFRC in terms of video quality by measuring the peak signal-to-noise ratio (PSNR) values of the encoded video at the sender-end and the PSNR values of the decoded video at the receiver-end. We also take the difference between the PSNR of the encoded video prior to sending at the sender-end and the PSNR of the decoded video at the receiver-end. The results are shown in TABLE III. The breakdown of the Y PSNR for each frame for Container and Foreman at the sender-end is shown in Fig. 7 and Fig. 8 respectively.

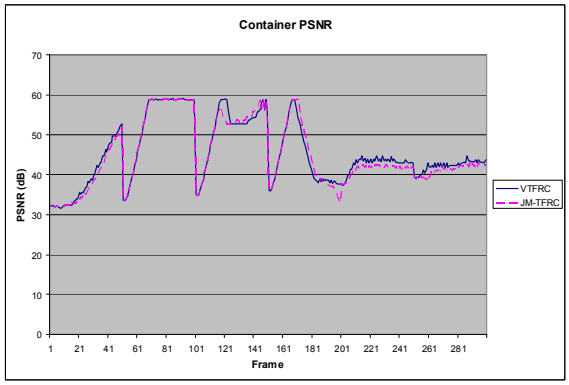

Fig. 7. The Y-PSNR of each frame for Container at the sender.

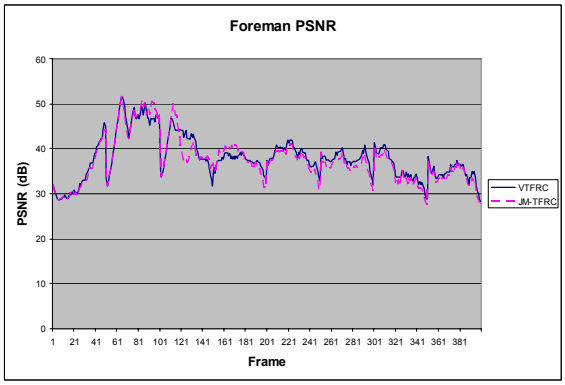

Fig. 8. The Y-PSNR of each frame for Container at the sender.

TABLE III. PSNR COMPARISONS

Sequence	VTFRC Mean Y-PSNR (dB)	JM-TFRC Mean Y-PSNR (dB)	PSNR Gain (dB)
Sender-End			
Container	45.49	44.97	0.52
Foreman	37.94	37.36	0.58
Grandma	45.29	44.31	0.98
Mother-Daughter	44.06	42.91	1.15
Salesman	45.45	44.73	0.72
Mean Overall PSNR Gain:			0.79
Receiver-End			
Container	43.37	42.25	1.12
Foreman	35.38	34.02	1.35
Grandma	43.37	43.02	0.35
Mother-Daughter	42.46	41.26	1.20
Salesman	41.36	39.01	2.34
Mean Overall PSNR Gain:			1.27
PSNR Difference between Sender and Receiver ends (Sender-end PSNR – Receiver-end PSNR)			
Container	2.12	2.72	-
Foreman	2.56	3.34	-
Grandma	1.92	1.29	-
Mother-Daughter	1.6	1.65	-
Salesman	4.09	5.72	-

From the sender-end results, it is shown that even though VTFRC encodes a frame at a higher rate opportunistically, it *still produced an overall mean PSNR gain of 0.79dB* over JM-TFRC. And even after losing video packets due to congestion, from the receiver-end results, VTFRC had an overall mean PSNR gain of 1.27dB over JM-TFRC.

The PSNR difference results show that even though VTFRC is in general more aggressive than JM-TFRC, *it does not contribute much to the network congestion* as the differences for VTFRC are comparable to that of JM-TFRC.

IV. CONCLUSION

This paper has proposed a joint source rate control and congestion control method that could opportunistically code video frames depending on the frame complexity and network conditions as interpreted by the TFRC protocol. Since only the TCP-friendly rate region is exploited to do this, the resulting flow is still TCP-friendly although it is slightly more aggressive. However, the video quality gains and the comparable congestion control justify this extra aggressiveness.

ACKNOWLEDGMENT

We thank Guillaume Jourjon from NICTA for his assistance in this work.

REFERENCES

[1] M. Handley, S. Floyd, J. Pahdye, and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," IETF, Request For Comments 3448, Jan. 2003.

[2] Widmer, J.; Denda, R.; Mauve, M., "A survey on TCP-friendly congestion control," *Network, IEEE* , vol.15, no.3, pp.28-37, May 2001

[3] S. Floyd and K. Fall. "Promoting the use of end-to-end congestion control in the Internet". *IEEE/ACM Transactions on Networking*, 7(4):458–472, 1999.

[4] Sally Floyd, Mark Handley, Jitendra Padhye, and Joerg Widmer, "Equation-Based Congestion Control for Unicast Applications", *SIGCOMM*, August 2000.

[5] J. Vieron and C. Guillemot, "Real-time constrained TCP-compatible rate control for video over the Internet," *IEEE Trans. Multimedia*, vol. 6, no. 3, pp. 634–646, Aug. 2004.

[6] Y.-G. Kim, J. Kim, and C.-C Jay Kuo, "TCP-friendly Internet video with smooth and fast rate adaptation and network-aware error control," *IEEE Trans. Circuit Syst. Video Technol.*, vol. 14, no. 2, pp. 256–268, Feb. 2004.

[7] Shih, C.H.; Wang, J.Y.; Shieh, C.K.; Hwang, W.S., "An integrated rate control scheme for TCP-friendly MPEG-4 video transmission," *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on* , vol., no., pp. 2124-2127 Vol. 3, 23-26 May 2005

[8] Peng Zhu; Wenjun Zeng; Chunwen Li, "Joint Design of Source Rate Control and QoS-Aware Congestion Control for Video Streaming Over the Internet," *Multimedia, IEEE Transactions on* , vol.9, no.2, pp.366-376, Feb. 2007

[9] Lee H. J., Chiang T., Zhang Y. Q.: Scalable rate control for MPEG-4 video. *IEEE Trans. Circuits, System and Video Tech*, Vol.10, Iss.6, Sep 2000, 878-894

[10] Li Z. G., Pan F., Lim K. P., Feng, G. N., Lin X., Rahardja S.: Adaptive Basic Unit Layer Rate Control for JVT. JVT-G012, 7th JVT Meeting, Pattaya, Thailand, March 2003

[11] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communications Review*, vol. 27, no. 1, Jan. 1997.

[12] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A Model Based TCP-Friendly Rate Control Protocol", *NOSSDAV'99*, 1999.

[13] Chung-Hsing Hsu and Ulrich Kremer, IPERF: A Framework for Automatic Construction of Performance Prediction Models, 1998.

[14] H.264/AVC JM Reference Software, http://iphome.hhi.de/suehring/tml/.