

Fuzzy Logic Congestion Control of Transcoded Video Streaming without Packet Loss Feedback

E. Jammeh, M. Fleury *Member, IEEE*, M. Ghanbari, *Fellow, IEEE*

Abstract

Congestion control of a variable bit-rate video stream crossing the Internet is crucial to ensuring the quality of the received video. When a fuzzy-logic congestion controller (FLC) changes the sending rate of a video transcoder, it does so without feedback of packet loss, using packet dispersion instead. Compared to the well-known TFRC and RAP controllers, the FLC's sending rate is significantly smoother, allowing it to more closely take-up available bandwidth at a bottleneck link. There is an accompanying order of magnitude reduction in packet losses. Due to better utilization of the available bandwidth, video quality is improved over time by several dBs in low packet loss conditions. The strength of the FLC solution is demonstrated by the resulting video quality when typical Web traffic forms the background traffic. The FLC avoids any risk of congestion collapse through fairness to coexisting TCP flows and is robust to changes in path delay and router buffer configuration.

Keywords: Video streaming, congestion control, fuzzy logic.

I. INTRODUCTION

Networked video communication [1] is achieved by determining the available bandwidth along an end-to-end path and adapting the encoded video rate accordingly. Volatile traffic load conditions on the Internet create the need for video-specific measures [2] that can accurately and in a timely manner determine the network state. Packet loss has traditionally signalled congestion to TCP, which has achieved remarkable success in avoiding excessive Internet congestion. Nevertheless, the limitation of packet loss has been identified as a performance bottleneck in TCP and enhancements to the protocol with a packet delay-based indicator have been proposed [3]. However, TCP emulators [4] for real-time transport over UDP (to avoid unbounded delivery delay through TCP) do include a packet loss factor in their models. While packet loss is acceptable in file transfer, as lost packets are simply re-transmitted through TCP's reliable transport, playout and decode deadlines must be met when streaming video. However, under congestion control of UDP, without avoidance measures, packet losses without retransmission do occur, degrading the delivered video quality. This paper shows that, because of reduced fluctuations in the sending rate, a fuzzy logic congestion controller (FLC) of video over UDP can more nearly approach an optimal regime, in which the available bandwidth is closely tracked with minimal packet loss. Input to the FLC is from a packet dispersion measure, which is a form of delay-based congestion control.

In [5], delay-based congestion control with the delay gradient was employed for applications such as video conferencing to arrive at low average end-to-end delay with minimal restriction of throughput. In the process it was found that output oscillations

Manuscript received May *, 2006; revised * *. This paper was recommended by Associate Editor Jianfei Cai.

The work was supported by the U.K. EPSRC under Grant EP/C538692/1.

The authors are with the University of Essex, Electronic Systems Engineering Department, Multimedia Network and Applications Group, Colchester, CO4 3SQ, United Kingdom (corresponding author: +44-1206-872817; fax: +44-1206-872900; e-mail: fleum@essex.ac.uk).

Copyright (c) 2007 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending an email to pubs-permissions@ieee.org.

were reduced along with delay variance. The method also avoided the ‘phase effect’ [6], whereby packet-loss probe-based congestion control introduces unfairness between streams across the same link, as the same stream may repeatedly suffer packet loss at the congested link. Congestion control of a video stream can be achieved through a rate-adaptive transcoder. Video transcoders, including ours [7] adapted for variable bit rate (VBR) streams, open up the possibility of sending a pre-encoded video bitstream at the maximum possible rate without overly exceeding the available network bandwidth. Hence, subsequent router buffering is able to cope with the output packet stream. In fact, it is quite possible to arrive at fewer packet losses or even avoid loss altogether by re-compressing an already compressed bitstream by means of a transcoder. Although in this paper we have applied fuzzy logic to a rate-adaptive transcoder, direct fuzzy logic control of the encoder quantizer step sizes is also possible. However, as pre-encoded video comprises the majority of video streams on the Internet, rather than live video streams, the description concentrates on transcoding.

Fuzzy logic, which has from its inception been extensively used for industrial and commercial control applications [8], is for us simply a convenient tool for handling un-modelled network congestion states. Within video coding it has found an application [9] in maintaining a constant video rate by varying the encoder quantization parameter according to the output buffer state, which is a complex control problem without an analytical solution. Fuzzy logic control of congestion is a sender-based system for unicast flows. The receiver returns a feedback message that indicates time-smoothed and normalized changes to packet inter-arrival time. These allow the sender to compute the network congestion level through pre-designed fuzzy models. The sender then applies a control signal to the transcoder’s quantization level, as a reflection of the anticipated congestion. Thus, congestion control without packet loss feedback is achieved by measuring packet stream dispersion arising when busy router queues are encountered, especially at tight links, representing the point of minimum available bandwidth on the network path. Fuzzy control is thus able to function in low packet loss environments.

A well-engineered FLC for transcoded video should:

- 1) Be TCP-friendly so that, in the event of proliferation of FLC streams within the Internet, there is a limited risk of congestion collapse.
- 2) Coexist with typical Internet traffic, consisting of long-term file transfer flows and short-term Web server connections.
- 3) Track the available bandwidth as closely as possible, though at the same time reducing or eliminating packet loss.
- 4) Achieve an optimally smooth stream, to avoid fluctuations in delivered video quality.

Items one and two are a measure of the quality of the solution, as without these stipulations a controller could simply greedily acquire bandwidth from other traffic. Items three and four are highly desirable for video traffic.

The remainder of the paper is organized as follows. The details of the system architecture and the FLC are given in Section II. Section III reports a set of simulation experiments. Finally, Section IV draws some conclusions, explaining why this paper proposes an FLC for video streams.

II. CONGESTION CONTROL

A. System Architecture

Fig. 1 shows a video streaming architecture in which fuzzy logic is utilized to control the bitrate. A video transcoder at the server [7] is necessary for pre-encoded video-rate adaptation. The client-side timer unit monitors the dispersion of incoming packets and relays this information to the congestion level determination (CLD) unit. The CLD unit monitors the outgoing packet stream, especially the packet sizes, and combines this information with feedback from the client, as a basis for determining the network congestion level, C_L . This unit also computes the congestion-level rate of change, δC_L . The timer unit measures the arriving packet inter-packet gaps (IPG's) before finding a time-smoothed and normalized estimate of the packet dispersion. An IPG is the time duration between the receipt of the end of one packet and the arrival of the next. The FLC takes C_L and δC_L , as inputs, and computes a sending rate that reflects the network's state. The appropriate change in the transcoder quantization level is then calculated. Transported packets are received by the client, de-packetized, decoded and displayed at video rate.

At the server, the video transcoder inputs the pre-encoded video and reduces its bit-rate in response to the control signal from the FLC. The lower bound to the sending rate was set to be 10% of the input sending rate. For the average input sending rate of 2 Mb/s in the simulations in Section III, a lower limit of 200 kb/s is sufficient for an acceptable video quality. The transcoded video is packetized, with one slice per packet, and sent across the network within a UDP packet. Apart from error resilience due to decoder synchronization markers, per-slice packetization also reduces delay at the server. Transcoded video packets are subsequently output with a constant IPG at the point of transmission. Ensuring a constant IPG reduces packet inter-arrival jitter at the client and also renders the streamed video more robust to error bursts.

B. Congestion Level Determination

In this study, the level of network congestion is measured by the application itself rather than by an auxiliary program such as [10]. The difference between reception from and transmission rate into the network, for the same application, is indicative of the network congestion level. In terms of packets, this amounts to finding packet dispersion, which is calculated from IPGs. The sending and receiving rates are measured from IPGs at the server and client respectively. The IPGs are then normalized by the packet size. At the server, the normalized IPG is the reciprocal of the bandwidth expended by the application in transmitting individual packets into the network, and a similar relationship applies at the client. Let the IPG of the packets be T_S and T_C for the packet entering the network and exiting the network, respectively. T_C will equal T_S when the available network bandwidth is equal to or more than the sending rate of the packets. A congested network will generally have a dispersive effect on the IPG, resulting in T_C being greater than T_S . The difference between T_C and T_S is, therefore, a graded measure of network congestion.

Normalizing the IPGs by the packet sizes makes them only dependent on the network congestion level. Knowing the normalized values of T_C and T_S will then enable computation of the network congestion level. Therefore, congestion level is determined without relying on packet loss, which is vital for video. An average packet transfer time G_a^S is measured at the

server, and G_a^C is measured for the client. The averages are measured in a frame transmission duration and it is these averages that are then used to determine the level of network congestion. The sending rate is changed when and only when a feedback message arrives at the receiver. The sending rate itself is only changed at the beginning of a video frame to ensure consistent quality within a frame. The following is a description of the network congestion-level measuring algorithm.

For two consecutive packets, of sizes S_n and S_{n-1} , received at times T_n and T_{n-1} , the transfer time, G_n , in general is defined as follows:

$$G_n = \frac{T_n - T_{n-1}}{S_n}, \quad (1)$$

where T_n is the arrival time of the current packet, T_{n-1} is that of the previous packet, and S_n is the size of the current packet. A set of transfer times, $\{G_w^S\}$ with cardinality sn , is measured at the server during a frame transmission duration. A set of transfer times at the client, $\{G_w^C\}$ with cardinality cn , is also collected over the same measurement period:

$$\{G_w^S\} = \{G_{s1}, G_{s2}, \dots, G_{sn}\}, \quad (2)$$

$$\{G_w^C\} = \{G_{c1}, G_{c2}, \dots, G_{cn}\}. \quad (3)$$

In general, $sn \neq cn$, because of the possibility of packet loss.

The members of sets $\{G_w^S\}$ and $\{G_w^C\}$ are assigned respectively to N_S and N_C equally-spaced bins. The range of the bins is found dynamically at the server and client to give:

$$N_S = \frac{G_{sn}^{max} - G_{sn}^{min}}{\delta r}, \quad N_C = \frac{G_{cn}^{max} - G_{cn}^{min}}{\delta r}, \quad (4)$$

where G_{sn}^{max} and G_{sn}^{min} are respectively the maximum and minimum of the $\{G_w^S\}$ (and similarly for the client). $1/\delta r$ is a pre-set resolution bitrate, which depends on the maximum encoding rate, with the number of bins in practice being restricted to 256. The intention of the binning procedure is to reduce the impact of IPG compression, which can occur in the presence of cross traffic [11] and can result in a spread of the measured transfer times. In [11], as capacity measurements are known to be multi-modal, the histogram width was set at a fraction of the interquartile range of pre-measurements, according to statistical practice. As pre-measurements of the range were not available in our system, based on potential cross traffic, δr was estimated to be equivalent to 1% of the encoding rate. If the resulting number of bins is too large or too small then the available bandwidth mode would not be as accurately estimated, owing to the spread in measurements [12].

From the binned time intervals, a frequency weighted average transfer time is calculated for the server and the client respectively as G_a^S and G_a^C in (5):

$$G_a^S = \frac{\sum_{i=1}^{N_S} V_i^S \times O_i^S}{\sum_{i=1}^{N_S} O_i^S}, \quad G_a^C = \frac{\sum_{i=1}^{N_C} V_i^C \times O_i^C}{\sum_{i=1}^{N_C} O_i^C}, \quad (5)$$

where V_i^S and V_i^C are the bin transfer values at the server and client, respectively, and O_i^S and O_i^C are their frequencies of occurrence.

A continuous exponentially weighted moving average filters out measurement noise from both values by updating G_{av}^S and G_{av}^C :

$$G_{av}^S \Leftarrow \alpha G_a^S + (1 - \alpha)G_{av}^S, \quad G_{av}^C \Leftarrow \alpha G_a^C + (1 - \alpha)G_{av}^C, \quad (6)$$

where at time zero G_{av}^S and G_{av}^C are set to the initial values of G_a^S and G_a^C respectively and the exponential parameter $\alpha \leq 1$ is typically set to 0.1, as higher values result in excessive fluctuations of the rate. The sending rate of the application into the network and the receiving rate of the client from the network can now be calculated as R_S and R_C in (7).

$$R_S = \frac{1}{G_{av}^S}, \quad R_C = \frac{1}{G_{av}^C}. \quad (7)$$

The difference between the two rates, R_D , can then be calculated:

$$R_D = R_S - R_C. \quad (8)$$

The network congestion level, C_L , can subsequently be calculated as:

$$C_L = \frac{R_D}{R_S} = 1 - \frac{G_{av}^S}{G_{av}^C}. \quad (9)$$

Finally, δC_L is also calculated as simply the difference between the present and previous value of C_L . Both C_L and δC_L are based on packet dispersion and these form the inputs to the congestion controller.

C. Fuzzy Logic Congestion Controller Design

Fuzzy logic emulates a control process, *as if* a human expert were regulating the transmission rate. Multiple fuzzy membership functions model the uncertainty in that expert's perception of the feedback, whereas an output rate decision is made precise by the process of defuzzification, which translates uncertainty in the output to a crisp value, i.e. a specific control signal value.

Fig. 2 is a block diagram of an FLC. Fuzzifiers convert the inputs C_L and δC_L into suitable linguistic variables. A knowledge base encapsulates expert knowledge of the application with the required control goals. It defines the labels that help specify a set of linguistic rules. The inference engine block is the intelligence of the controller, with the capability of emulating the human decision making process, based on fuzzy logic, by means of the knowledge database and embedded rules for making those decisions. Lastly, the defuzzification block converts the inferred fuzzy control decisions from the inference engine to a crisp value, which is converted to a control signal, CT in Fig. 2, to the transcoder, which then outputs a re-compressed bitstream.

Fuzzification is the term given to the application of a membership function, μ , to a data value to find its membership possibility, i.e. $\mu(x)$ yields the possibility of membership of the fuzzy subset for which μ is the membership function. The input variables were fuzzified by means of triangular-shaped membership functions, being the usual compromise between reduced computation time at the expense of a sharper transition from one state to another. Choosing the number of membership functions is important, as it determines the smoothness of the bit-rate granularity. However, the number of membership functions is directly proportional to the computation time. The congestion level, the rate at which it changes, and the control output

were each partitioned into a set of overlapping triangular membership functions, with the overlap such that extent of any one triangle reached the midpoint of the base of another.

Table I defines the linguistic variables for inputs C_L and δC_L . The inference rules are in the same form as the following examples:

if C_L is L and δC_L is NVH then S is SPL

if C_L is M and δC_L is NVH then S is SZ ,

where S is the defuzzified output from the controller. L , M , and NVH are input membership functions for the related fuzzy subsets and SPL and SZ are the output membership functions. A given pair of values of C_L and δC_L may each be possible members of two fuzzy subsets with the same or differing possibilities and, therefore, more than one rule may apply, as in the examples, when C_L may be in the fuzzy subsets associated with L and M . The defuzzified output has the same linguistic variables as δC_L in Table I, but abbreviated with a prefix of S, e.g. SZ (zero). All the inference rules of a complete set used by us in the simulations of Section III are given in Table II. The tabulated format of Table II captures in a concise form the information contained in English sentences constrained in the manner of the two previous examples.

The center of gravity method is used for defuzzification. Equation (10) maps the input to the output of the controller:

$$CT = \frac{\sum_{i=1}^M X_i K_i}{\sum_{i=1}^M K_i} \quad (10)$$

where M is the number of rules, X_i is the value of the output for rule i , and K_i is the inferred weight of the i^{th} output membership function. More specifically, X_i is the value at the middle of the range of data values that are possible members of the i^{th} fuzzy subset. K_i is the area under the i^{th} output membership function, clipped by the minimum possibility (which may be zero) of membership of C_L and δC_L in the two input membership functions of the i^{th} rule. As more than one rule may well apply, (10) is in the form of a weighted average of the values arising from the different rules that are applicable, with outputs of zero for those remaining fuzzy subsets for which the data value is not a member. From Table II, a typical number of rules, M , is 45 (total number of outputs). A few different rules map to the same outputs.

The control signal CT , as specified in Eqn. (10), is normalized to the range $(0, 1]$, subject to a minimum lower bound. For input bitrate R_{in} , the target output bitrate is $R_{out} = CT.R_{in}$ through multiplication. In steady state, to achieve sending rate R_{out} , the quantization scale of the transcoder is directly proportional to CT and the dynamic range of available quantizers. In order to manage the combined transcoder and target decoder buffer occupancy [13] without increasing delay, a correction factor is applied in a picture dependent manner. As CT equates to α in Algorithm A of [13], the interested reader is referred to that paper for further details.

III. EXPERIMENTS

The algorithm was simulated with the well-known ns-2 network simulator (version 2.30 used). The simulated network, with a typical ‘dumbbell’ topology, had a bottleneck link between two routers and all side link bandwidths were provisioned such that congestion would only occur at the bottleneck link. That is access links from the senders and to the receivers were set

to 100 Mb/s. The default buffer size of the bottleneck link routers was configured to be twice the bandwidth delay product, as is normal in such experiments to avoid packet losses from too small a buffer. The one-way delay of the bottleneck link was set to 5 ms and the side links' delays were set to 1 ms. The bottleneck link routers' queuing policy was defaulted to be FIFO (drop-tail). Random Early Detection (RED) routers rather than drop-tail are reported [14] as difficult to configure in a simulation in such a way that the behavior is uniform across a range of background traffic intensities. Though it is expected that RED will improve congestion control if and when it is widely deployed, [14] reports that for lightly loaded links there is a danger of error bursts if RED is applied.

In all experiments, when under fuzzy control, the IPG at the sender was set to 2.2 ms, which corresponds to the video packetization characteristics of Section II-A. Network-state decisions were fed back from the receiver to the fuzzy controller after every frame or frame transmission interval (e.g. 40 ms). An MPEG-2 video 'news clip' with moderate motion and Group of Picture (GOP) structure of $N=12$, $M=3$ was selected, originally pre-encoded at an average rate of 2.0 Mb/s.

Comparison was made with the TCP-friendly Rate Control (TFRC) protocol, the subject of an RFC [15] and a prominent method of congestion control from the originators of the 'TCP-friendly' concept. To ensure fairness the publicly available TFRC ns-2 simulator model¹ (in the form of `object Tcl` scripts to drive the simulator) was employed. In TFRC, the sending rate is made a function of the measured packet loss rate during a single round-trip-time (RTT) duration measured at the receiver. The sender then calculates the sending rate according to the TCP throughput equation given in [16]. In the TFRC experiments, a TFRC controller dispatched fixed-size UDP packets across the same network tight link, varying the IPG according to the available bandwidth, as estimated by the TFRC feedback mechanism. In the TCP throughput equation employed by TFRC, notice that the packet length explicitly appears as a linear scaling factor, allowing TFRC to adjust its behavior according to a constant packet size. Packet loss and RTT appear as non-linear factors in the equation. In [17], it was found that the median packet length for UDP streaming was 640 B, which is similar to the fixed 700 B packet length of TFRC in these tests.

Additionally, comparison was made with RAP [14], which is an alternative to equation-based modeling. RAP varies the IPG between fixed-size packets to allow its average sending rate to approach TCP's for a given available bandwidth. Every smoothed RTT, RAP implements an Arithmetic Increase Multiplicative Decrease (AIMD)-like algorithm [18], with the same thresholds and increments as TCP. Because this would otherwise result in TCP's 'sawtooth'-like rate curve, with obvious disruption to multimedia streams, RAP introduces fine-grained smoothing (turned on in this paper's tests), which takes into account short- and long-term RTT trends. Because of its pioneering role and its close resemblance to TCP, RAP has frequently served as a point of comparison for congestion controllers. To ensure fairness to RAP public ns-2 models² were availed of.

Internet measurement studies [19] have demonstrated a typical Internet traffic mix to consist of longer term flows, 'Tortoises', representing file transfers, and transient HTTP connections, 'Dragonflies'. In the set of experiments herein, one FLC video source and up to ten TCP sources were passed across the link. The first five TCP sources were 'dragonflies' with a random duration of between one and five seconds. These sources were generated from a uniform distribution and with an off duration of between one and five seconds, also randomly generated from a uniform distribution. The remaining five TCP sources were

¹This is obtainable from <http://www.icir.org/tfrc/>

²These are obtainable from <http://netweb.usc.edu/reza/RAP/NewRAP/>

configured as ‘tortoise’, with an on duration of between five and twenty seconds and an off duration between one and five seconds, all also randomly generated from a uniform distribution. In the first experiment, only one TCP source was present as background traffic, in the second two TCP sources acted as background traffic and so on, and all ten TCP sources were on as background traffic for a tenth experiment. All experiments were repeated ten times with different seeds and the mean result was taken.

The bottleneck link capacity was set to 500 kb/s in Fig. 3, which records the mean quality of the received news clip video, for the range of traffic intensities. The plots for the sets of tests (independently conducted for FLC, TFRC and RAP) show an improved trend resulting from the FLC. The TFRC PSNR is poorer, owing to greater packet losses and less efficient utilization of the available bandwidth. Mean standard deviations (s.d.’s) of the PSNR over time and averaged over the set of tests for the FLC and TFRC were respectively 1.5 and 4.4 dB, indicating that delivered PSNR by the FLC was considerably smoother over time than that of TFRC. The RAP behavior was broadly similar to TFRC. Though certainly TFRC improves upon the video quality arising from RAP, with the RAP mean PSNR s.d. being 6.5 dB, both are inferior to the FLC video quality.

An examination of the time-wise behavior of the congestion controllers gives an insight into the cause of Fig. 3’s results. The available bandwidth was altered by changing over time a simulated constant bit rate cross traffic. Fig. 4 shows that an FLC is capable of a more accurate rate adaptation to changing available bandwidth than TFRC. This leads to the lower loss rates, as reported in Fig. 5. Notice that the scale of losses is of the order of thirty times larger than that of the FLC, the vertical axis being logarithmic. Moreover, the level that the transcoder output would need to be offset to avoid all loss would actually be closer to the available bandwidth than that achievable by TFRC, because of TFRC’s less accurate rate adaptation. As TFRC employs fixed size packets, the FLC packets were also made to be of fixed size (700 B) between rate decision instances (after every frame transmission). As slices can potentially span packet boundaries, there is some loss in synchronization at the decoder but the comparison is made closer between the two controllers. Also plotted in Fig. 5 is the packet loss response of the RAP [14] under the same experimental conditions. As Fig. 5 indicates, RAP’s packet losses are even worse than TFRC and an order of magnitude greater than those with the FLC.

The relative effect on video quality for the first 250 frames of the video transmitted in Fig. 4 is captured in Fig. 6 (a) for TFRC and FLC. The transition at about 200 pictures represents the stepped drop in available bandwidth between 5 and 10 s. The FLC maintains a consistent quality (s.d. 1.2 dB), whereas the high TFRC loss rate contributes to the repeated quality drops (s.d. 4.7 dB). Moreover at the transition in available bandwidth, the FLC induced PSNR more quickly adapts, whereas TFRC’s adjustment displays a lag. Therefore, the FLC is better able to adjust to changes in available bandwidth in low loss (for the FLC) conditions. During the 250 frame period, the average FLC PSNR was 37.4 dB, 3 dB above that of TFRC. The spurts in quality attainable through the TFRC’s direct transmission of the pre-encoded video are unsustainable and would lead to disturbing fluctuations in quality. In Fig. 6 (b) the FLC PSNR is compared to that of RAP. At the transition around 200 frames, similar remarks apply to RAP to those for TFRC, with RAP’s mean PSNR being 34 dB (s.d. 5.0 dB).

To ensure that the FLC could react to a variety of scenarios, a set of tests very similar to those in [20] for TCP-friendliness was also conducted. As in [20], link delay was set to 50 ms and the tight link capacity was initially set to 2.5 Mb/s. The

same number of flows, FLC VBR video and FTP TCP (New Reno variety) sessions, shared the link. Because of the lower bound on the transcoder's bit rate, for experiments with 40 and 80 flows, the link capacity was increased to 4 Mb/s. The router buffer queue size was set to two, four, and eight times the bandwidth delay product. This allows assessment of the impact of queuing delay over the network path on feedback messages. Fig. 7 reports the experiments showing that the TCP-friendliness of the FLC actually increases as more flows are introduced. The FLC's TCP-friendliness is in the same margins as for [20] and confirmed to be close to an ideal figure of one.

The effect of varying delay and buffer configurations was also examined. In Fig 8 (a) the link delay is set to a range of different values. The buffer size is set to twice the delay-bandwidth product, for fixed bandwidth capacity of 1 Mb/s. For link delays over 75 ms, there is a loss in responsiveness caused by the delayed feedback. This will affect the positioning of servers. If a buffer's size is not set to catch all possible burst sizes (if its size is below the delay-bandwidth product) then increased packet loss will occur. This will affect the averaging ability of the congestion estimation mechanism. As a result, in Fig 8 (b) with the same bandwidth capacity, for a small fixed buffer size, there is a further loss in responsiveness, though this is more serious for delays over 75 ms.

IV. CONCLUSIONS

Congestion control combined with rate-adaptive transcoding is a convenient way to provide a video service across the Internet, configured as it presently is without uniform QoS deployment. Established congestion controllers for real-time flows are strongly motivated by the need to avoid congestion collapse and, as such, mimic with UDP the average throughput behavior of TCP. This form of congestion control is not especially suited to video streaming, because of the need for packet loss feedback and because of residual TCP-like rate fluctuations. The work reported in this paper set out to meet the numbered points listed in Section I, preserving the qualities of existing congestion controllers such as TFRC and RAP, but exceeding their capability in delivering a smooth flow and reducing packet loss to minimal levels. This goal was achieved by means of packet dispersion feedback to a Fuzzy Logic Congestion Controller (FLC).

In a typical experiment, the packet loss of an FLC video stream was well below and the bitrate standard deviation over time was reduced compared to both TFRC and RAP. The mean received video quality in tests was consequently several dB above that resulting from TFRC. Smoother flows allow graceful degradation of transmitted video in the face of congestion and smaller receiver buffers. There is also a reduced risk of packet loss at router queues. Crucially, through transcoder control, the rate can be reduced so that there is almost no loss but, through the smoothness of the sending rate, the available bandwidth is more closely approachable than less smooth controllers such as RAP and TFRC. The FLC's ability to track available bandwidth is also relatively unaffected by different network path delays and buffer sizes. The end result will translate into a better subjective experience by the user viewing the received video and reduce the problem of power-hungry buffering memory on mobile devices.

REFERENCES

- [1] D. Wu, Y. T. Hou, W. Zhu, Y.-Q. Zhang, and J. H. Peha, "Streaming video over the Internet: Approaches and directions," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 282–299, March 2001.
- [2] T. Turetti and C. Huitema, "Videoconferencing on the Internet," *IEEE/ACM Transactions on Networking*, vol. 4, no. 3, pp. 340–351, June 1996.
- [3] J. Martin, A. Nilsson, and I. Rhee, "Delay-based congestion avoidance for TCP," *IEEE/ACM Transactions on Networking*, vol. 11, no. 3, pp. 356–363, June 2003.
- [4] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, vol. 15, no. 3, pp. 28–37, May/June 2001.
- [5] W. Dabbous, "Analysis of delay based congestion avoidance algorithm," in *High-Performance Networking IV*, 1992, pp. 283–298.
- [6] S. Floyd and V. Jacobsen, "Traffic phase effects in packet-switched gateways," *Computer Communications Review*, vol. 21, no. 2, pp. 26–42, 1991.
- [7] P. A. A. Assunção and M. Ghanbari, "A frequency domain video transcoder for dynamic bit-rate reduction of MPEG-2 bit streams," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 8, pp. 953–967, December 1998.
- [8] H. Takagi, "Application of neural networks and fuzzy logic to consumer products," *IEEE Technology Updates Series: Fuzzy Logic Technology and Applications*, vol. 1, pp. 8–12, 1994.
- [9] P. M. Grant, Y.-S. Saw, and J. M. Hannah, "Fuzzy rule based MPEG video rate prediction and control," in *Eurasip ECASP Conference*, June 1997, pp. 211–214.
- [10] M. Jain and C. Dovrolis, "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Transactions on Networking*, vol. 11, no. 4, pp. 537–549, August 2003.
- [11] C. Dovrolis, P. Ramanathan, and D. Moore, "Packet dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Transactions on Networking*, vol. 12, no. 6, pp. 963–977, December 2004.
- [12] —, "What do packet dispersion techniques measure?" in *IEEE INFOCOM*, April 2001, pp. 905–910.
- [13] P. A. A. Assunção and M. Ghanbari, "Buffer analysis and control in CBR video transcoding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 83–92, February 2000.
- [14] R. Rejaie, M. Handley, and D. Estrin, "RAP: An end-to-end rate-based congestion control mechanism for realtime streams in the Internet," in *IEEE INFOCOM*, vol. 3, March 1999, pp. 1337–1345.
- [15] M. Handley, S. Floyd, J. Padyhe, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification," 2003, IETF RFC 3448.
- [16] J. Padyhe, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation," in *ACM SIGCOMM '98*, September 1998, pp. 303–314.
- [17] J. Chun, Y. Zhu, and M. Claypool, "FairPlayer or FoulPlayer? —head to head performance of RealPlayer streaming video over UDP versus TCP," Worcester Polytechnic Institution, MA, Tech. Rep., May 2002.
- [18] L. Cai, X. Shen, J. Pan, and J. W. Mark, "Performance analysis of TCP-friendly AIMD algorithms for multimedia applications," *IEEE Transactions on Multimedia*, vol. 7, no. 2, pp. 339–335, April 2005.
- [19] N. Brownlee and K. C. Claffy, "Understanding Internet traffic streams: Dragonflies and tortoises," *IEEE Communications*, vol. 40, no. 10, pp. 110–117, October 2002.
- [20] Y.-G. Kim, J. W. Kim, and C.-C. J. Kuo, "TCP-friendly Internet video with smooth and fast rate adaptation and network-aware error control," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no. 2, pp. 256–268, February 2004.

TABLE I
EXAMPLE FLC LINGUISTIC VARIABLES FOR INPUTS C_L AND δC_L .

C_L		δC_L	
Value	Meaning	Value	Meaning
L	Low	NVH	Negative very low
M	Medium	NH	Negative high
H	High	NM	Negative medium
VH	Very high	NL	Negative low
EH	Extremely high	Z	Zero
		PL	Positive low
		PM	Positive medium
		PH	Positive high
		PVH	Positive very high

TABLE II
A COMPLETE SET OF FUZZY-LOGIC INFERENCE RULES

$\delta C_L \setminus C_L$	L	M	H	VH	EH
NVH	SPH	SPM	SPL	SZ	SNL
NH	SPM	SPL	SZ	SNL	SNM
NM	SPL	SZ	SZ	SNM	SNM
NL	SPL	SZ	SNL	SNM	SNH
Z	SZ	SNL	SNM	SNH	SNH
PL	SNL	SNL	SNM	SNH	SNH
PM	SNL	SNM	SNH	SNH	SNVH
PH	SNM	SNH	SNH	SNVH	SNVH
PVH	SNM	SNH	SNVH	SNVH	SNVH

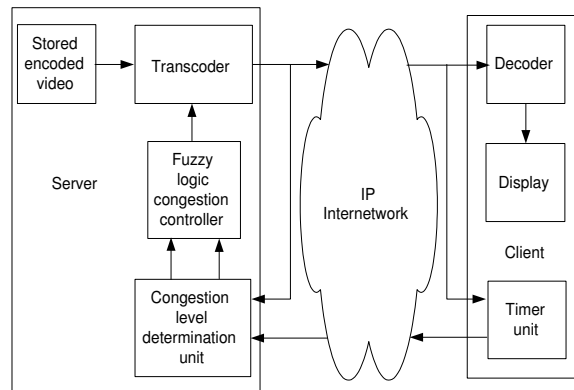


Fig. 1. Block diagram of video-streaming architecture

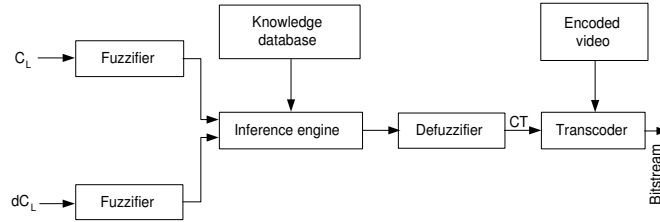


Fig. 2. Block diagram of a fuzzy-logic congestion controller

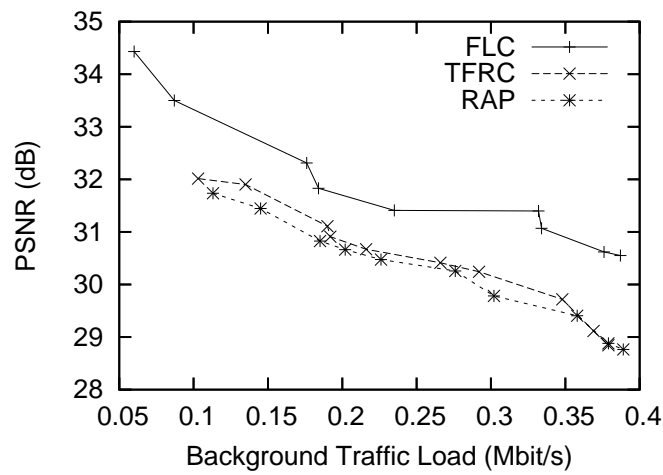


Fig. 3. Received mean Y-PSNR of a VBR video stream with an increasing background traffic load for a 500 kb/s bottleneck bandwidth using either FLC, TFRC, or RAP.

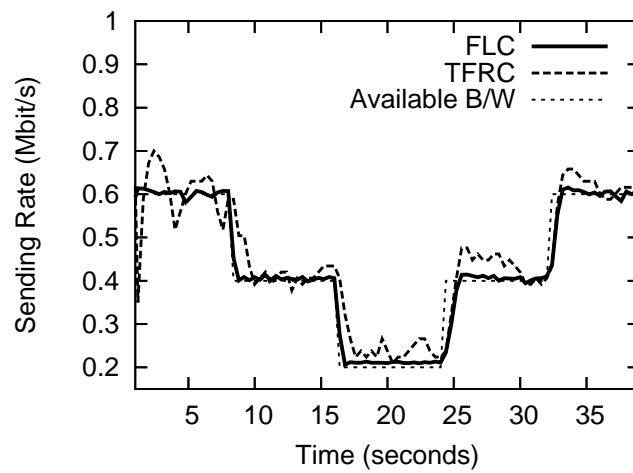


Fig. 4. Tracking a changing available bandwidth with FLC VBR video or TFRC sending rates.

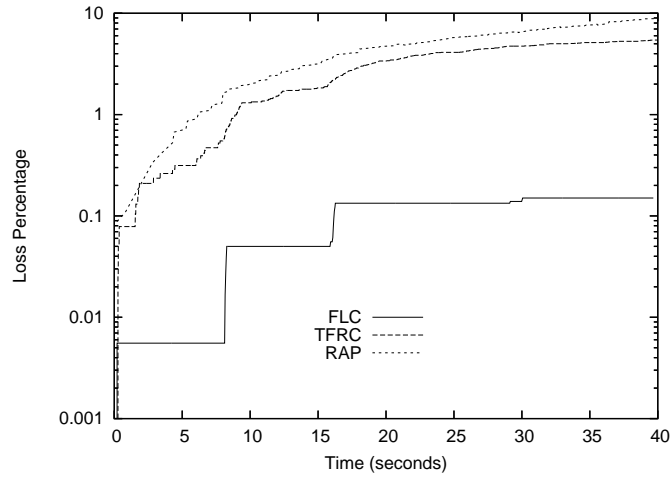
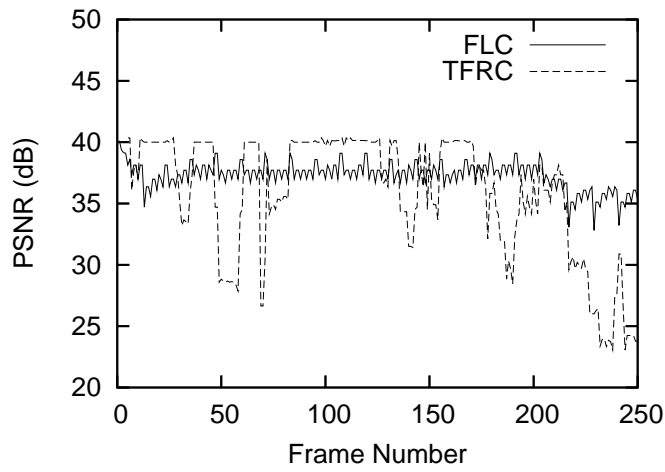
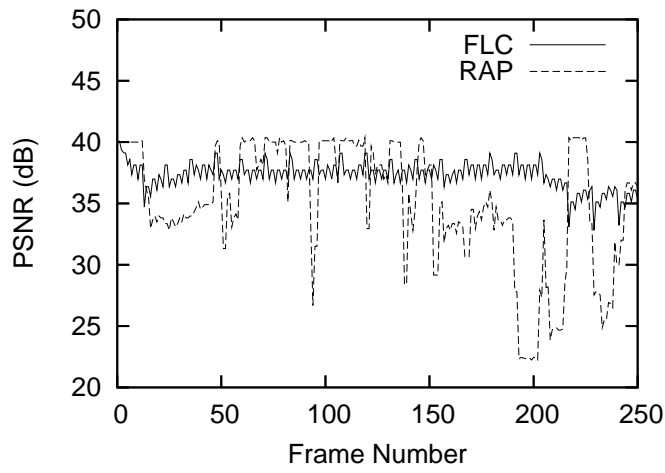


Fig. 5. Packet loss rate for the sending rates of Fig. 4 under TFRC or FLC VBR video.



(a)



(b)

Fig. 6. Received Y-PSNR of a VBR sequence for the first 250 frames after the transmission of Fig. 4 under a) TFRC or FLC b) RAP or FLC.

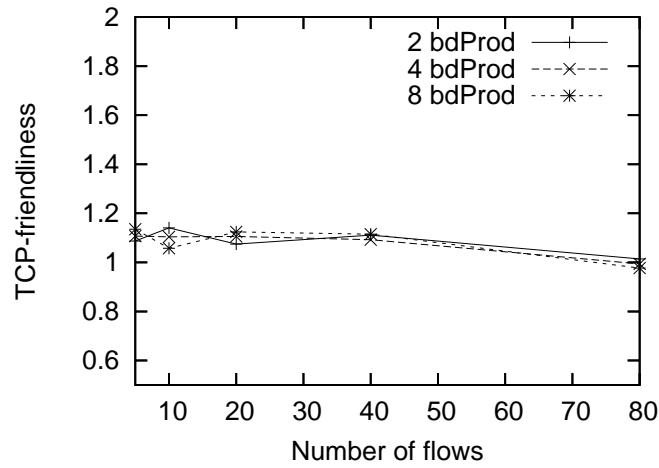


Fig. 7. TCP-friendliness of FLC for multiple competing TCP flows and two, four and eight times the bandwidth-delay product size queues.

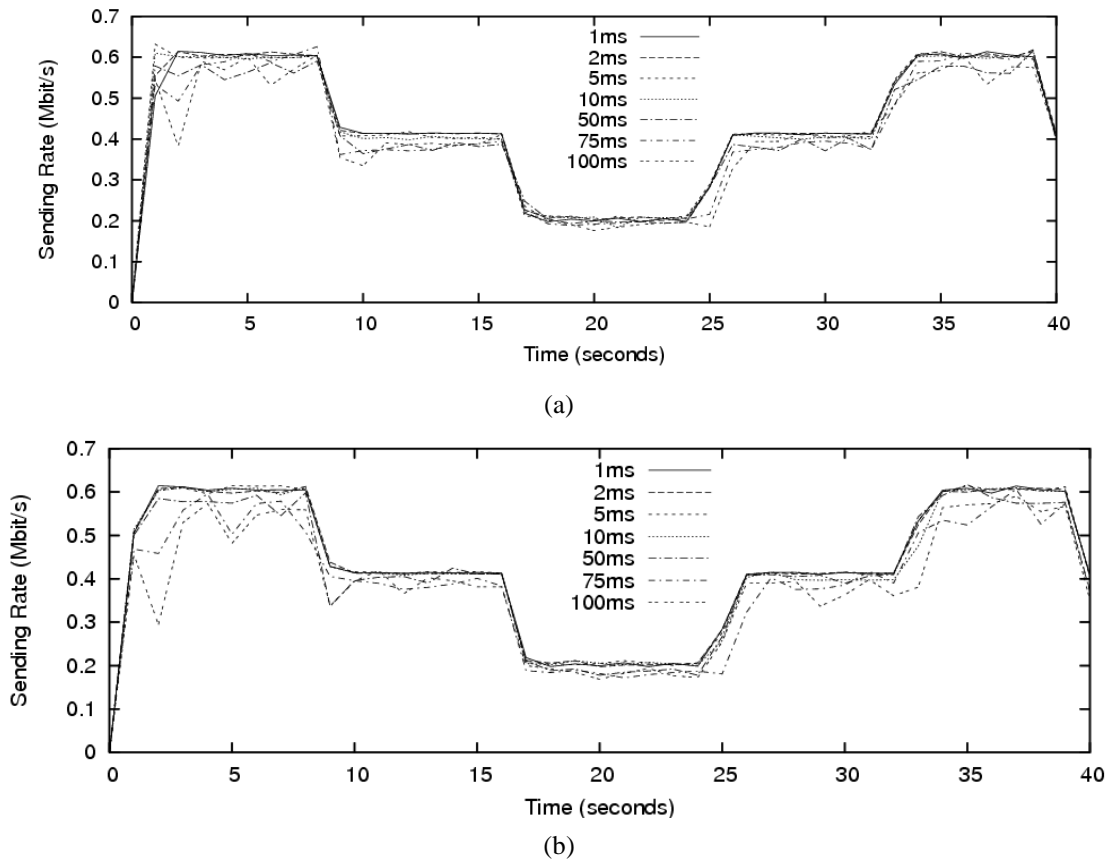


Fig. 8. Tracking a changing available bandwidth for a 1 Mb/s capacity link with FLC VBR video for different values of link delay with buffer size set to (a) twice the delay-bandwidth product (b) the delay-bandwidth product with a fixed 1 ms delay.

FOOTNOTES

Manuscript received May *, 2006; revised August *, 2007. This paper was recommended by Associate Editor Jianfei Cai.

The work was supported by the U.K. EPSRC under Grant EP/C538692/1.

The authors are with the University of Essex, Electronic Systems Engineering Department, Multimedia Network and Applications Group, Colchester, CO4 3SQ, United Kingdom (corresponding author: +44-1206-872817; fax: +44-1206-872900; e-mail: fleum@essex.ac.uk).

This is obtainable from <http://www.icir.org/tfrc/>

These are obtainable from <http://netweb.usc.edu/reza/RAP/NewRAP/>