Rate-adaptive Video Streaming through Packet Dispersion Feedback

E. A. Jammeh, M. Fleury, and M. Ghanbari

University of Essex,

Department of Computing and Electronic Systems,

Wivenhoe Park, Colchester, CO4 3SQ, United Kingdom

tel: +44 - 1206 - 872817

fax: +44 - 1206 - 872900

e-mail {eajamm,fleum,ghan}@essex.ac.uk

Abstract

The anticipated growth of IPTV makes selection of suitable congestion controllers for video stream traffic of vital concern. Measurements of packet dispersion at the receiver provide a graded way of estimating congestion, which is particularly suited to video as it does not rely on packet loss. The paper presents a closed loop congestion controller, which dynamically adapts the bitstream output of a transcoder or video encoder to a rate less likely to lead to packet loss. The video congestion controller is based on fuzzy logic, with packet dispersion and its rate of change forming the inputs. Compared to TCP-emulators such as TFRC and RAP, which rely on packet loss for real-time congestion control, the fuzzy logic trained system's sending rate is significantly smoother when multiple video-bearing sources share a tight link. Using a packet dispersion method similarly results in a fairer allocation of bandwidth than TFRC and RAP. These gains for video traffic are primarily due to better estimation of network congestion through packet dispersion but also result from accurate interpretation by the fuzzy logic controller.

1 Introduction

This paper introduces packet dispersion as a measure of congestion more suited as a control input for rate adaptive transcoded video streams than traditional means. Packet dispersion is the spacing of two or more packets originally sent back-to-back [1]. Essentially, packet dispersion measures router buffer queueing delay, as dispersion is caused by intervening packets from cross-traffic sources. The TCP method of gauging congestion in the wireline Internet [2] is through packet loss, with round-trip time (RTT) also playing a role. Some packet loss is acceptable in file transfer, as lost packets can simply be re-transmitted. However, in streaming video, continual retransmission until a lost packet arrives is not possible, as playout and decode deadlines must be met. For that reason, video streaming is usually accomplished through unreliable UDP transport under application layer congestion control, rather than reliable TCP transport. However, if no retransmissions are permitted then measures need to be taken to avoid packet loss. This is because packet losses are hostile to fragile encoded video streams because their effect through predictive motion estimation persists over multiple frames, until the next purely spatially encoded frame. In fact, though packet losses certainly indicate congestion they do not provide direct information on the degree or level of network congestion or conversely the available network bandwidth.

On the other hand, packet dispersion measured at the receiver obviously removes the dependency on packet loss and also avoids the problem of asymmetric forward and return Internet paths inherent in RTT measurements, which require complex averaging and smoothing algorithms. Applying a packet dispersion-based congestion controller has certainly proved advantageous from a video-centric viewpoint, as this paper makes clear. Compared to TCP-emulators [3] such as TFRC [4] and RAP [5] the sending rate is significantly smoother in scenarios where there are multiple video-bearing sources present. And using a packet dispersion method results in a fairer allocation of bandwidth than TFRC and RAP in the presence of other controlled UDP streams sharing a tight link. Though the proposed controller could be applied to live video encoding, given the predominance of pre-encoded video streams in the Internet, it is more likely to be configured with a transcoder.

Video transcoders [6] 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. A perceived weakness of transcoding, a lack of scalability, has not proved a weakness in practice [7] and by acting in the frequency domain [8], the impact on latency is minimized.

In the proposed system, the receiver returns a feedback message that indicates time-smoothed and normalized changes to packet inter-arrival times (packet dispersion). These allow the sender to compute the network congestion level. The sender then applies a control signal to the quantization level of the transcoder (for pre-encoded video) or to that of a live encoder, as a reflection of the anticipated congestion. Thus, congestion control 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. This is realized through the non-linear characteristics of fuzzy logic control (FLC), with packet dispersion and its rate of change forming the inputs. This logic should cope with delayed and imprecise feedback messages [9], which may return over the same or part of the outgoing network path, interacting with the forward traffic.

The main question explored in this paper is whether real-time congestion controllers such as RAP and the later TFRC are actually suited to video traffic or could a packet dispersion based controller perform equally or better. For video streaming across fixed networks, the trend has been towards large playout buffers at the receiver, with typical start-up delay reported as 5-15 s in [10], which at 25 frame/s represents at least 125 frames. Therefore, to simplify analysis overflow at the receiver buffer is discounted. During the build-up of congestion, which is evidenced by increased packet delay, limited or no packet loss occurs. Undoubtedly, at the onset of full-blown congestion packet loss will occur. On these occasions, a useful addition to the input may also be packet loss, which in an FLC might be introduced in a modular fashion. Nevertheless, to simplify the analysis this presentation concentrates on packet dispersion as an input. Section 2 considers the development of real-time TCP-emulators, while Section 3 is a detailed description of how a packet dispersion based controller works. Section 4 presents our results, showing that: 1) certainly a packet-dispersion-based fuzzy controller is fully able to track available bandwidth; and 2) when there is only other video stream cross-traffic it is particulary effective. Finally, Section 5 sums up the paper.

2 Background

Congestion control of real-time streams can be based on emulating the averaged throughput of a TCP source, in order to participate in TCP's collective congestion control policy. The TCP throughput equation presented in [11] is reproduced as (1), since it is employed by TCP-Friendly Rate Control (TFRC) [12], which forms a basis of comparisons in Section 4. In

$$T(t_{RTT}, t_{RTO}, s, p) = \text{min}\left(\frac{w_m.s}{t_{RTT}}, \frac{s}{t_{RTT}\sqrt{\frac{2bp}{3}} + t_{RTO}\text{min}\left(1, 3\sqrt{\frac{3bp}{8}}\right)p(1+32p^2)}\right), \tag{1}$$

where s is the segment size (TCP unit of output), t_{RTT} is the round-trip time (RTT), p is the packet loss rate, b is the number of packets acknowledged by each ACK, t_{RTO} is TCP's retransmission timeout value and w_m is the maximum window size. Equation (1) models TCP Reno, while no known analytic form exists for the behaviour of TCP Selective Acknowledgement (SACK) [12], and to the best of our knowledge for later versions of TCP.

Probe-based rate control [13] tests the available bandwidth through ACK-reported packet loss. One characteristic shared by rate-based methods is the aim to reproduce the average behaviour of TCP's Additive Increase Multiplicative Decrease (AIMD) congestion avoidance algorithm, to impart fairness amongst coexisting flows [14]. The Rate Adaptation Protocol (RAP) [5], also forming a basis of comparisons in Section 4.3, and [15] are examples of this approach. In Loss-Delay Based Adaption Algorithm (LDA+) [16], the AIMD increase and decrease parameters are adjusted by probing the available bandwidth through measurement of the inter-packet gap (IPG) of two packets sent out back-to-back. Many measurement techniques calculate either available bandwidth or capacity from packet inter-arrival times, for example [17]. These schemes rely on the network imposing some structure on the distribution of packet inter-arrival times, from which the network bandwidth characteristics can be inferred. When adapted to congestion control, they provide a link with our method described in the next Section.

3 Congestion Control through Packet Dispersion

3.1 System Architecture

Fig. 1 shows the video streaming architecture modeled in our experiments. A video transcoder at the server is necessary for pre-encoded video-rate adaptation, while a video decoder at the client decodes the received video stream in real time. The in-house frequency-domain video transcoder reported in [8] was employed in this paper's experiments and applied to MPEG-2 encoded video streams. This transcoder obtains a new quantizer scale through a linear rate-quantization dependency. If R^G is the given or input bit-rate and R^T is the target bit-rate, a new quantization scale, $\hat{q_k}$ is obtained through:

$$\hat{q_k} = \frac{R_i^T}{R_i^G} q_k + q_m \frac{R_i^G - R_i^T}{R_i^G},\ k \in S_i,\ i \in P, \tag{2}$$

with $1 \le \hat{q_k} \le 31$. The maximum quantization scale $q_m = 31$. P is the set of slices in the coded bitstream, while $S_i,\ i \in P$ are the sets of coded macroblocks in each slice. Thus, (2) shows the calculation of the new quantization scale $\hat{q_k}$ for the k^{th} macroblock S_i. This method of re-quantization was selected for practicality and is an approximation, though it should be noted that a more accurate model is possible through rate-quantization curves [18]. For on-line video clip streaming, it is the video encoder's rate that is adapted. Algorithms for rate adaptation in the spatial domain for the standard codecs are documented in [19] and in the standards documents themselves.

The client-side timer unit monitors the dispersion of incoming packets and relays this information to the congestion level determination unit. The congestion level determination module monitors the outgoing packet stream, especially the packet sizes, and combines this information with the receiver device feedback, as a basis for determining the network congestion level, C_L. This unit also computes the congestion-level rate of change, δC_L. An IPG is the time duration between the receipt of the end of one packet and the arrival of the next. Principally, the timer unit measures the arriving IPGs before finding a time-smoothed and normalized estimate of the packet dispersion. 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 (or video encoder) 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 further reduces its bit rate in response to a control signal from the FLC. The lower bound to the sending rate was set to be 10% of the input sending rate. For the approximate input sending rate of 2 Mb/s in the simulations in Section 4, a lower limit of 200 kb/s is sufficient for an acceptable video quality. Transcoded video packets are subsequently output with a constant IPG at the point of transmission. Ensuring a constant IPG reduces the packet inter-arrival jitter at the client and also renders the streamed video more robust to error bursts.

3.2 Congestion Level Determination

Available path bandwidth can be taken as a measure of the level of network congestion. Several schemes, for example [20], [21], have been proposed to estimate the available network bandwidth directly from packet inter-arrival times, but these schemes, which do not currently work from within the application, take a relatively long time to arrive an accurate estimate.

In this study, the level of network congestion is measured by the application itself rather than by an auxiliary program. 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.

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. In other words, as far as the application sending the packets is concerned, equality will apply when the network is not congested. Though it is theoretically possible that no intervening packets from other flows will insert themselves in router buffers even when congestion occurs, this is not a problem for practical controllers. Congestion occurs when multiple flows arrive coincidentally with the video flow and insert themselves at intervening positions in a router buffer, thus causing dispersion. A congested network will, therefore, 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 measure of network congestion. The more congested the network, the more is the difference. The IPGs T_C and T_S, apart from being dependent on the sending rate of the packets, are dependent on two variables: 1) the level of network congestion; and 2) the size of the packets.

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. Thus, 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 these averages are then used to measure the level of network congestion. The sending rate is changed when and only when a feedback message arrives at the sender. The sending rate itself is only changed at the beginning of a video frame to ensure consistent quality within a frame. Depending on its coding type each frame will contribute a different average bit-rate. If a frame occurs at a shot or scene boundary the bit-rate may change considerably. Therefore, it is wise to update (consider changing) the bit-rate at every frame instance to avoid the risk of packet loss. Less frequent feedback will not alter the accuracy of the congestion level estimate, as a sample is taken from every received network packet to form a running average. However, less frequent feedback will reduce the sensitivity of the system to network fluctuations. The frequency of feedback does not necessarily result in network adaptation if no significant change to the congestion level has occurred.

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:

$$G_n = \frac{T_n - T_{n-1}}{S_n}, \tag{3}$$

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:

$$\{G_w^S\} = \{G_{s1}, G_{s2}, \ldots G_{sn}\} \tag{4}$$

$$\{G_w^C\} = \{G_{c1}, G_{c2}, \ldots G_{cn}\}. \tag{5}$$

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:

$$N_S = \frac{G_{sn}^{max} - G_{sn}^{min}}{\delta r}, \tag{6}$$

$$N_C = \frac{G_{cn}^{max} - G_{cn}^{min}}{\delta r}, \tag{7}$$

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 [17] and can result in a spread of the measured transfer times. In [17], 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, δr was estimated, based on potential cross traffic, 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 [22].

Based on 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 (8) and (9):

$$G_a^S = \frac{\sum_{i=1}^{N_S} V_i^S \times O_i^S}{\sum_{i=1}^{N} O_i^S} \tag{8}$$

$$G_a^C = \frac{\sum_{i=1}^{N_C} V_i^C \times O_i^C}{\sum_{i=1}^{N} O_i^C}, \tag{9}$$

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:

$$G_{n,av}^S = \alpha G_{n,a}^S + (1-\alpha) G_{n-1,av}^S \tag{10}$$

$$G_{n,av}^C = \alpha G_{n,a}^C + (1-\alpha) G_{n-1,av}^C, \tag{11}$$

where $G_{0,av}^S$ and $G_{0,av}^C$ are set at time zero 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 packet reception instance is indexed by n in (10) and (11). 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 respectively in (12) and (13).

$$R_S = \frac{1}{G_{av}^S}. \tag{12}$$

$$R_C = \frac{1}{G_{av}^C}. \tag{13}$$

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

$$R_D = R_S - R_C. \tag{14}$$

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

$$C_L = \frac{R_D}{R_S}, \tag{15}$$

$$C_L = 1 - \frac{G_{av}^S}{G_{av}^C}. \tag{16}$$

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.

3.3 Fuzzy Logic Congestion Controller

The purpose of congestion control is to adapt the transmission rate to the available network bandwidth with a reduced rate variability, maximal utilization of the available bandwidth, and improved delivered video quality. Fig. 2 is a block diagram of an FLC. A fuzzification unit converts the inputs C_L and δC_L, based on packet dispersion, into suitable linguistic variables. A fuzzy set is expressed as a set of rules which take the form of linguistic expressions. These rules express experience of tuning the controller and, in the methodology, are captured in a knowledge database. A knowledge base encapsulates expert knowledge of the application with the required control goals. It defines the labels that help specify the set of linguistic rules. The inference engine block is the intelligence of the controller, with the capability of emulating the human decision making process, using fuzzy logic, by means of the knowledge database and embedded rules for making those decisions. Lastly, the defuzzification block converts inferred fuzzy control decisions from the inference engine to a crisp value, which is converted to control signal, CT in Fig. 2, to the transcoder, which then outputs a re-compressed bitstream.

In a fuzzy subset, each member is an ordered pair, with the first element of the pair being a member of a set S and the second element being the possibility, in the interval $[0, 1]$, that the member is in the fuzzy subset. This should be compared with a Boolean subset in which every member of a set S is a member of the subset with probability taken from the set $\{0, 1\}$, in which a probability of 1 represents certain membership and 0 represents non-membership.

As a simple example, in a fuzzy subset of (say) ‘tall’, the possibility that a person with a given height taken from the set S of heights may be called tall is modeled by a membership function, which is the mapping between a data value and possible membership of the subset. Notice that a member of one fuzzy subset can be a member of another fuzzy subset with the same or a different possibility. Membership functions may be combined according to a set of ‘if $\ldots$ then’ rules to make inferences such as `if x is tall and y is old then z is happy`, in which `tall`, `old` and `happy` are membership functions of the matching fuzzy subsets and `x, y, z` are linguistic variables (names for known data values).

In practice, the membership functions are applied to the data values to find the possibility of membership of a fuzzy subset and the possibilities are subsequently combined through defuzzification to provide a precise output. We have applied a semi-manual method of deriving the rules, combining human knowledge of protocol and network behavior with testing by simulator. The fuzzy model behavior itself was examined through Matlab Fuzzy Toolbox v. 2.2.4. This results in a widely applicable but static set of rules. The FLC's behavior can be predicted from its output surface, formed by knowledge of its rule table and the method of defuzzification.

Table 1 defines the linguistic variables for inputs C_L and δC_L. The defuzzified output has the same linguistic variables as δC_L in Table 1, 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 4 are given in Table 2. The centre of gravity method was used for defuzzification Equation (17) maps the input to the output of the controller:

$$CT = \frac{\sum_{i=1}^{M} X_i K_i}{\sum_{i=1}^{M} K_i} \tag{17}$$

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, (17) 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. The membership functions for the two inputs and selection of the output value are illustrated in Fig. 3.

The control signal CT, as specified in Eqn. (17), 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 [23] without increasing delay, a correction factor is applied in a picture dependent manner. As CT equates to α in Algorithm A of [23], the interested reader is referred to that paper for further details.

4 Experiments

4.1 Network Simulation

The algorithm was simulated with the well-known ns-2 network simulator (version 2.28 used). The simulated network, with a typical 'dumbbell' topology, Fig. 4, had a tight link between two routers and all side link bandwidths were provisioned such that congestion would only occur at the tight link. The one-way delay of the tight link was set to 5 ms and the side links' delays were set to 1 ms. The tight link's queueing policy was defaulted to be droptail or FIFO and the queue size was set to twice the bandwidth-delay product, as is normal in such experiments to avoid packet losses from too small a buffer. In all experiments, when under fuzzy control, the IPG at the sender was set to 2.2 ms, which is equivalent to a European Standard Input Format (SIF) MPEG-2 encoded video stream with 18 slice/frame and one slice in each UDP packet.[1] Network-state decisions were fed back from the receiver to the fuzzy controller after every frame or frame transmission interval (*e.g.* 40 ms for 25 frame/s).

Table 3 summarizes the characteristics of three MPEG-2 encoded video clips employed in tests. It is normal in the video coding community to employ test clips of less than 10 s. The generic video clips we use are longer than this to allow characteristic behaviour to be tracked over sufficient frames. In general, an encoded bitrate varies in a cyclic manner due to the GOP structure, though shot and scene changes will cause the form to vary over time. 1000 frames of 40 s duration are sufficient and efficient in large-scale tests as a way of establishing the ability to control a bit-rate varying in this way. The main test video, 'News clip', consists of a standing presenter with a changing background with moderate motion. Additionally, a 'BBC childrens TV programme (Blue Peter) with moving presenter and animations, and a 'Football match excerpt, the latter with markedly

[1]Each slice consists of a row of 22 macroblocks. Headers can be inserted at the start of slices to ensure decoder synchronization at this level. Per-slice packetization reduces delay at the server.

more motion than the other two clips were used as a point of comparison. The output for all three was at a Variable Bit Rate (VBR).

4.2 Tracking Bandwidth

The tracking ability of several controllers is now reported. Comparison was made with the TCP-friendly Rate Control (TFRC) protocol, the subject of an RFC [12] and a prominent method of congestion control from the originators of the 'TCP-friendly' concept [24]. A conformant non-TCP flow is TCP-friendly "when it does not reduce long-term throughput of any coexistent TCP flow more than another TCP flow" under the same conditions. The publicly available TFRC ns-2 simulator model[2] (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 RTT duration measured at the receiver. The sender then calculates the sending rate according to the TCP throughput equation [11] given in Section 2.

RAP [5] is an alternative to equation-based modelling, varying 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 AIMD-like algorithm, with the same thresholds and increments as TCP. Because this would otherwise result in TCP's 'sawtooth'-like rate curve, with obvious disruption to video streams, RAP introduces fine-grained smoothing (turned on in this paper's tests), which takes into account short- and long-term RTT trends. RAP has a known weakness in heavy congestion [3], as it does not employ timeouts and, hence, is likely to be more aggressive than TCP. To ensure fairness to RAP publicly-available ns-2 models[3] were gratefully taken advantage of.

In both TFRC and RAP, the constant packet size was set in the supplied scripts to 1000 B, which is the default setting from the TFRC web site. TFRC's data source provided packets up to the available rate, with, as recommended in [12], the smoothing algorithm turned on in the simulation model. Decision frequency was every RTT. RAP's source was as for TFRC, though the RAP model assumes that rate transitions are quantized.

[2]This is obtainable from `http://www.icir.org/tfrc/`

[3]These are obtainable from `http://netweb.usc.edu/reza/RAP/NewRAP/`

Fig. 5 demonstrates that TFRC, RAP and the FLC are able to follow scheduled changes by the simulator. Leaving aside RAP for the moment, it is apparent that FLC achieves similar behaviour without the need for packet loss feedback, whereas TFRC (and RAP) relies on packet loss to be able to track the scheduled changes. Therefore, the main message of Fig. 5 is that packet dispersion is a more favourable means of tracking bandwidth in respect to video. Including an offset or safety margin from the predicted sending rate of TFRC and FLC would still enable FLC to follow the available bandwidth but would result in TFRC losing its ability to follow closely the scheduled changes. In TFRC's case offsetting the predicted rate would cause it to once again probe until packet loss caused a reduced (and offset) rate to be applied. In FLC's case, adjustment to the model to achieve the offset reduces the output sending rate when it is close to the available bandwidth without otherwise changing its behaviour.

In Fig. 5, RAP has excessive rate fluctuations. The main determinant of RAP's behaviour, as decision frequency is similar to TFRC, appears to be the size of (stepped) rate transitions, which coarsens the transition levels available, despite 'fine-grain smoothing'. The RAP fluctuations imply that a reduction to the predicted output rate to reduce loss would need to be greater, decreasing the bitrate available for encoded video and, hence, reducing the delivered quality. Offsetting the RAP predicted rate would result in RAP losing its ability to closely track available bandwidth and lead to a greater depth in the sending rate fluctuations. The negative subjective experience of a changing delivered video quality arising through fluctuations in the encoded rate should also be taken into account.

Further experiments were performed to allow the effect of different packet sizes to be judged. In [25] a trace taken from the Ames Internet exchange, found that the modal packet size values were: 40 B (2.69 billion packets) (TCP ACK packet size); 576 B (514 million packets)(minimum size guaranteed to be transferred by all routers without fragmentation — much used by web and mail servers) and 1500 B (1.49 billion) packets (maximum Ethernet packet size). In Fig. 6, the controller's tracking ability is confirmed for a range of maximum packet sizes approaching Ethernet frame size. As the FLC varies its packet size according to the available bandwidth, for fixed IPG, it is only by changing the maximum that the effect of differing packet sizes can be judged. For a maximum packet size of 200 B, the output is simply unable to take advantage of the higher available bandwidths, which explains the lower plot at this packet size. In general, a notable feature of the our controller is that its tracking ability is relatively unaffected by changes in its packet size.

4.3 Inter-protocol Response

The ability of each controller to coexist with traffic controlled by a different congestion controller (inter-protocol) was considered. The scenario envisaged would occur if a server was controlling multiple streams each with their congestion controller as part of an IPTV or video-on-demand application.

In Fig. 7, the average bandwidth occupied by the TFRC flows is always above that of the FLC flows and above the ideal or equally-shared bottleneck bandwidth. This behaviour of TFRC, when exposed to a tight link without TCP traffic, would work to the disadvantage of other traffic. If replicated in a network, it is also likely that this behaviour would result in increased packet loss for the TFRC flows. It is postulated that the cause is over-sensitivity on the part of TFRC to variation in packet loss and RTT. In general, the result of Fig. 7 is an indication that the FLC acts as the better statistical multiplexor.

Though the FLC has equivalent or superior performance, nevertheless, if the bottleneck constraint is such that an equally allocated share to the video service cannot deliver an acceptable video quality then the allocation ought to be adjusted in favour of video. For example, at a certain level of congestion, if and when the video share falls below a lower bound for acceptable delivery, then higher weight ought to be given to video, and an equal share distribution be restored when congestion eases, a policy and mechanism that is called multimedia-friendliness in [26]. However, for a transcoded video service the input video can be 2 Mb/s, its lowest output bit rate can be 200 kb/s, and its lower bound can be reduced still further to as low as 20 kb/s through spatio-temporal transcoding [27]. In fact 20 k/s is sufficient for sub-QCIF (176×144 PAL pixel resolution) pictures at low frame rates. Within such a very large dynamic range, share adjustment may never be required.

4.4 Delivered Video quality

Internet measurement studies [28] have demonstrated a typical Internet traffic mix to consist of longer term flows, ‘Tortoises’, representing file transfers, and transient HTTP connections, ‘Dragonflies’. In the experiments of Fig. 8, 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 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. 8, which records the mean quality of the received news clip video in terms of luminance Peak Signal-to-Noise Ration (Y-PSNR), for the range of traffic intensities. Within the simulator, a record is kept of which packets of the encoded video bearing stream are lost. This data is then removed from a compressed video stream fed to a reference MPEG-2 decoder. The result of decoding this stream compared to that of the same stream without packet loss is used to calculate the PSNR on a frame-by-frame basis, as is normal. 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, 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, both are inferior to the FLC video quality.

The performance of the FLC was also tested against the three different video clips introduced in Section 4.1. In Fig. 9, the background traffic load characteristics were the same as those for Fig. 8, as was the bottleneck capacity. In the case of the 'BBC' and 'News' clips, reasonable video quality is achieved in the presence of one background flow and the transcoder reduces the rate as more background sources are turned on. The greater complexity of 'Football' presents a problem, as the quality is already poor with just one background flow, explaining why there is limited subsequent quality degradation. The packet losses rates at the differing background loads were similar for all three clips, whatever their quality, indicating that the reduction in quality is mainly due to rate reduction. Therefore, the FLC maintains its performance even though the packet size distributions are obviously different for each video clip.

4.5 Value of Dispersion Measure

To confirm the value of a delay-based measure such as dispersion a comparison was made between loss and delay measures. The set-up in Fig. 4 was replicated in a live testbed with two typical, medium-performance Cisco 2600 routers replacing the router and hub. A 2 Mbit/s serial link was placed between the routers. A video stream with average bit-rate of 187 kb/s was streamed across the link in the face of various cross-traffic. (The original video was CIF-sized with a 30 frame/s rate with moderate motion in an ”Interview” clip of 60s, chosen to avoid packet segmentation.) Cross-traffic patterns with a Normal (Norm) or Pareto (Par) probability density function (pdf) were injected into the link at mean rates of 1.5, 1.8 and 1.9 Mbit/s.

Table 5 records the packet loss rates for two different packet queue lengths. Interestingly, for Norm (1.9), Par (1.5), Par (1.8) and Par (1.9), the increase of the queue length (Ql) from 200 to 400 packets did not reduce the packet loss in the video stream but increased it. This phenomenon was observed during the experiments when the background traffic rate was higher than that of the video stream. The background traffic was able to acquire a greater number of empty packet buffer slots in the router queue, while the video stream was relatively neglected.

On the other hand for a delay-based measure, in Fig. 10, the left y-axes represent queueing delay, while the right y-axes represent the total sending rates presented to the serial link. The x-axes represent time during the streaming sessions. For Normal pdf cross-traffic, Fig. 10, the measured packet-by-packet packet delay of the video stream gradually rises to the maximum permitted by the given queue length. After the build up, the delay follows the bandwidth changes of the combined video- and cross-traffic. The time to reach the maximum delay varies based on the level of background traffic and the queue length: with cross-traffic of 1.8 Mbit/s, delay saturation is reached after 15 s and 20 s for respectively Ql = 200 and Ql = 400 packets. For bursty Pareto background traffic, the delay response tracks the peaks and lows present when the 2 Mbit/s threshold is crossed by the presented traffic. There is no obvious build up in delay, as occurred when the background traffic was less bursty. Because of the background packet burst pattern, the maximum delay is rarely reached. Nevertheless, packet-by-packet delay measurement has given a fine trace of what was occurring inside the router queue before the serial link. As packet dispersion is essentially an averaged and weighted measure of relative delay, one can conclude that it is an efficient estimate of congestion.

5 Conclusions

Congestion control combined with rate-controlled transcoding is a convenient way to provide an IPTV service across the Internet, configured as it presently is without uniform QoS deployment. Established congestion controllers are strongly motivated by the need to avoid congestion collapse and, as such, mimic with UDP the average throughput behaviour of TCP. It is an open question whether this form of congestion control is suited to video streaming, because of the need for packet loss feedback, when no loss is preferred for video transport, and because of the residual TCP-like rate fluctuations that persist. An interesting finding from the results in this paper is that the two main contending controllers perform similarly when tracking stepped bandwidth transitions with a single flow. However, when multiple flows from the same type of controller compete with each other or with flows from another controller, then dispersion-based FLC is superior in terms of rate smoothness and fairness to other traffic. Given that packet dispersion can function in conditions with low to no packet loss, its advantages for video-bearing flows seem clear.

While packet loss is a binary signal of congestion, packet dispersion provides a ‘multi-bit’ indication of congestion level. After normalization and smoothing over time, in this paper’s system packet dispersion acts as an input to a fuzzy logic controller. Packets forming a single video frame are a convenient unit over which to base measurement of dispersion. Other choices of dispersion-based controller are possible but fuzzy logic appears better to map onto the problem’s non-linear characteristics than by conventional modeling through an equation. Though the fuzzy logic controller is based on heuristics it can be tested as thoroughly as the equation-based technique. Freed of the necessity to mimic TCP's *modus operandi*, a video system actually proves more able to coexist with competing video traffic, avoiding the threat of induced congestion. Packet size is a significant issue as this will affect buffer occupancy. The results in this paper indicate that fuzzy logic control is comparatively unaffected by packet length dependencies. Type-2 fuzzy logic has recently gained a following as a way of increasing the range of uncertainty that fuzzy membership functions with fixed or crisp boundaries are not always able to capture. Therefore, fuzzy logic is inherently a method that can be adapted to the congestion control needs of future networks.

Acknowledgement

This work is supported by the U.K. EPSRC under Grant EP/C538692/1. We thank Dr M. Paredes Farrera for conducting some experiments for us.

References

[1] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and capacity estimation. *IEEE/ACM Transactions on Networking*, 10(1):12–26, 2001.

[2] W. Stevens. TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms, 1997. RFC 2001.

[3] J. Widmer, R. Denda, and M. Mauve. A survey on TCP-friendly congestion control. *IEEE Network*, 15(3):28–37, May/June 2001.

[4] M. Handley, S. Floyd, J. Padyhe, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification, 2003. IETF RFC 3448.

[5] 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*, volume 3, pages 1337–1345, March 1999.

[6] T. J. Shanableh and M. Ghanbari. Hybrid DCT/pixel domain architecture for heterogeneous video transcoding. *Signal Processing: Image Communication, Special Issue on Multimedia Adaptation*, 18(8):601–620, September 2003.

[7] H. Kasai, M. Nilsson, T. Jebb, M. Whybray, and H. Tominaga. The development of a multimedia transcoding system for mobile access to video conferencing. *IEICE Transactions on Communications*, 10(2):2171–2181, October 2002.

[8] 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*, 8(8):953–967, December 1998.

[9] C. E. Rohrs, R. A. Berry, and S. J. O'Halek. A control engineer's look at ATM congestion avoidance. In *IEEE Global Telecommunications Conference, GLOBECOM'95*, pages 1089–1094, 1995.

[10] E. Cianca, F. H. P. Fitzek, M. De Sanctis, M. Bonanno, R. Prasad, and M. Ruggieri. Improving performance for streaming video services over CDMA-based wireless networks. In *International Symposium in Wireless Personal Multimedia Communication*, September 2004.

[11] J. Padyhe, V. Firoiu, D. Towsley, and J. Krusoe. Modeling TCP throughput: A simple model and its empirical validation. In *ACM SIGCOMM '98*, pages 303–314, September 1998.

[12] M. Handley, S. Floyd, J. Padyhe, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification, 2003. IETF RFC 3448.

[13] D. Wu, Y. T. Hu, and Y.-Q. Zhang. Transporting real-time video over the Internet: Challenges and approaches. *Proceedings of the IEEE*, 88(12):1855–1875, 2000.

[14] L. Cai, X. Shen, J. Pan, and J. W. Mark. Performance analysis of TCP-friendly AIMD algorithms for multimedia applications. *IEEE Transactions on Multimedia*, 7(2):339–335, April 2005.

[15] D. Bansal and H. Balakrishnan. Binomial congestion control algorithms. In *IEEE INFOCOM*, volume 3, pages 631–640, April 2001.

[16] D. Sisalem and A. Wolisz. LDA+ TCP-friendly adaptation: A measurement and comparison study. In *10th International Workshop on Network and Operating Systems Support for Digital Audio and Video*, June 2000.

[17] C. Dovrolis, P. Ramanathan, and D. Moore. Packet dispersion techniques and a capacity-estimation methodology. *IEEE/ACM Transactions on Networking*, 12(6):963–977, December 2004.

[18] W. Ding and B. Liu. Rate control of MPEG video coding and recording by rate-quantization modeling. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(1):12–20, 1996.

[19] H. Sun, X. Chen, and T. Chiang. *Digital Video Transcoding for Transmission and Storage*. CRC Press, Boca Raton, FL, 2005.

[20] M. Jain and C. Dovrolis. End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput. *IEEE/ACM Transactions on Networking*, 11(4):537–549, August 2003.

[21] V. Ribeiro, R. Riedi, R. Barabanuik, J. Navratil, and L. Cottrell. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop (PAM)*, 2003.

[22] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *IEEE INFOCOM*, pages 905–910, April 2001.

[23] P. A. A. Assunção and M. Ghanbari. Buffer analysis and control in CBR video transcoding. *IEEE Transactions on Circuits and Systems for Video Technology*, 10(1):83–92, February 2000.

[24] 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, August 1999.

[25] C. Shannon, D. Moore, and K. C. Claffy. Beyond folklore: Observations on fragmented traffic. *IEEE/ACM Transactions on Networking*, 7(6):709–720, December 2002.

[26] C. Chen, Z. G. Li, and Y. C. Soh. TCP-friendly source adaptation for multimedia applications over the Internet. In *15th International PacketVideo workshop (PV2006)*, April 2006.

[27] T. Shanableh and M. Ghanbari. Heterogeneous video transcoding to lower spatio-temporal resolutions and different encoding formats. *IEEE Transactions on Multimedia*, 2(2):101–110, June 2000.

[28] N. Brownlee and K. C. Claffy. Understanding Internet traffic streams: Dragonflies and tortoises. *IEEE Communications*, 40:110–117, October 2002.

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 1: Example FLC linguistic variables for inputs C_L and δC_L.

$\delta C_l \backslash C_l$	*CL*	*CM*	*CH*	*CVH*	*CEH*
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

Table 2: Table showing a complete set of fuzzy-logic inference rules

Feature	Value
Size	352×288 pixel
Colour depth	24-bit
Number of frames	1000
Frame rate	25 f/s
Duration	40 s
GOP structure	n=12,m=3

Table 3: MPEG-2 video clip characteristics, GOP = Group of Pictures

Congestion controller	Bottleneck b/w (Mb/s)	Flow1 (F1) s.d. (Mb/s)	Flow2 (F2) s.d (Mb/s)	Abs(F1 - F2) s.d. (Mb/s)
TFRC	2.0	0.44849	0.63709	0.18860
RAP	2.0	1.35377	0.97761	0.37616
Fuzzy	2.0	**0.27979**	**0.28815**	**0.00836**
TFRC	5.0	1.48334	1.47669	0.00665
RAP	5.0	2.80038	2.13319	0.66719
Fuzzy	5.0	**1.38128**	**1.53012**	**0.14884**

Table 4: S.D.s for a sample of two flows over 120 s tests.

	Cross-traffic characteristics					
Ql	Norm 1.5	Norm 1.8	Norm 1.9	Par 1.5	Par 1.8	Par 1.9
200	0	2.30	4.67	2.58	6.95	7.26
400	0	2.16	4.87	2.68	6.99	7.39

Table 5: Packet loss rate for a constant IPG input video stream with Normal and Pareto pdf cross traffic at 1.5, 1.8 and 1.9 Mbit/s, with two packet queue lengths (Ql).

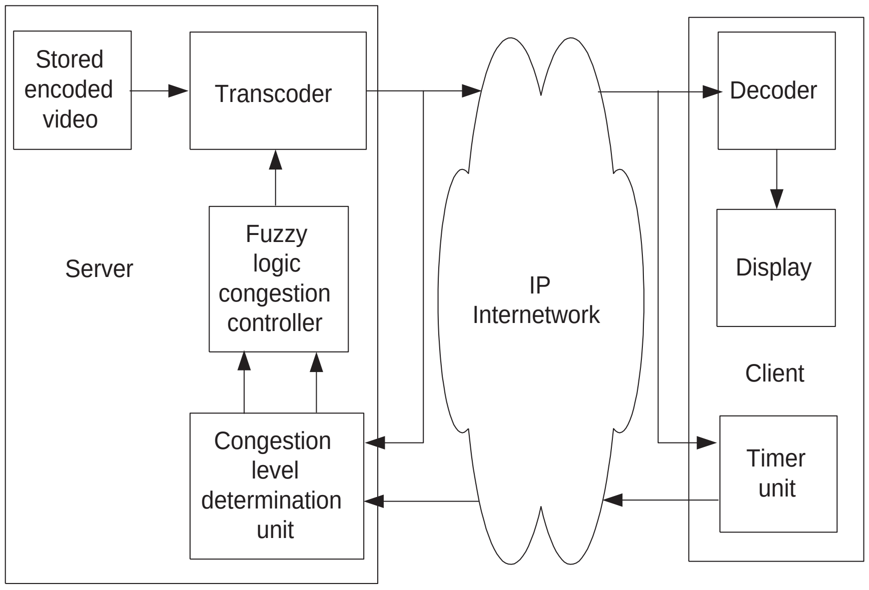


Figure 1: Block diagram of video-streaming architecture

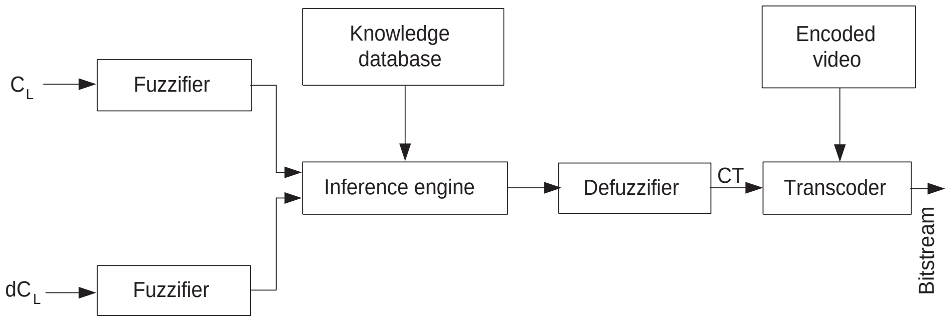


Figure 2: Block diagram of a fuzzy-logic congestion controller.

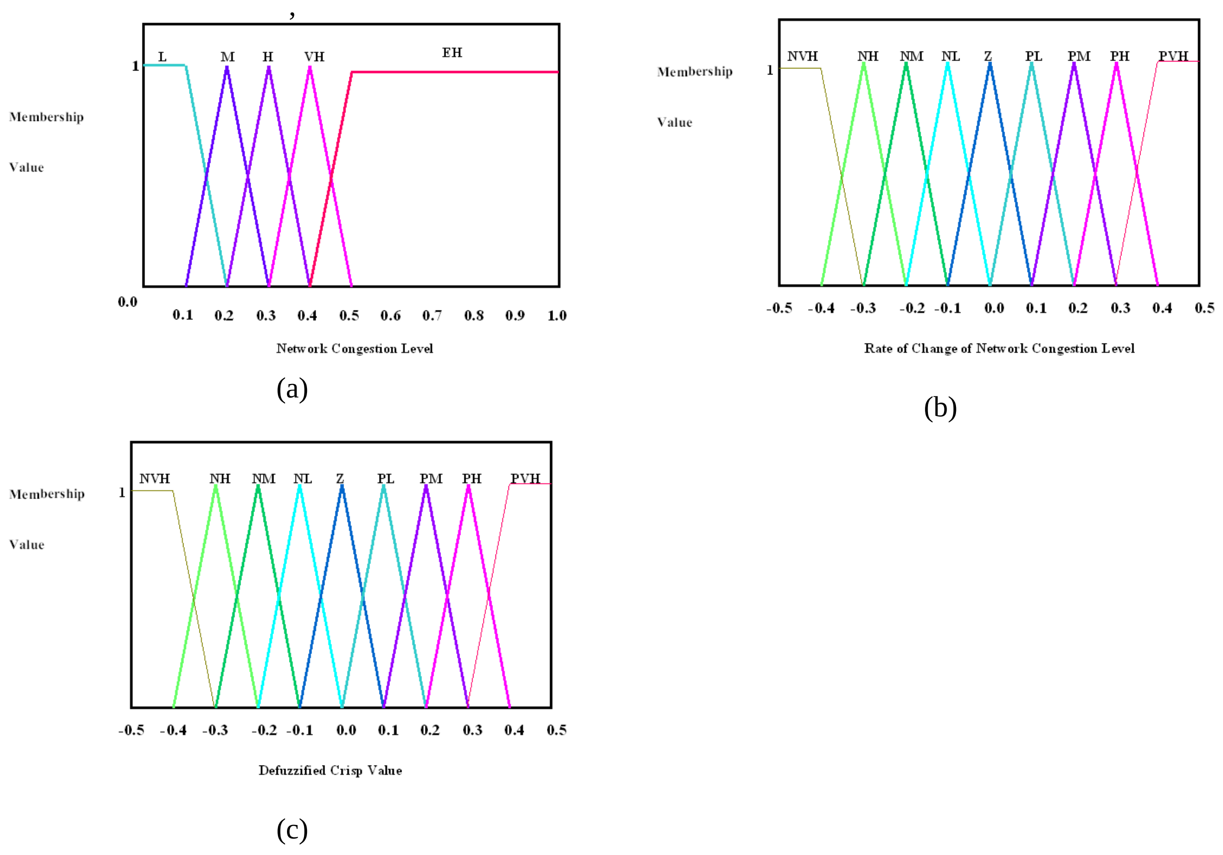


Figure 3: Fuzzy membership functions of a) Cl, b) δCl, c) output value.

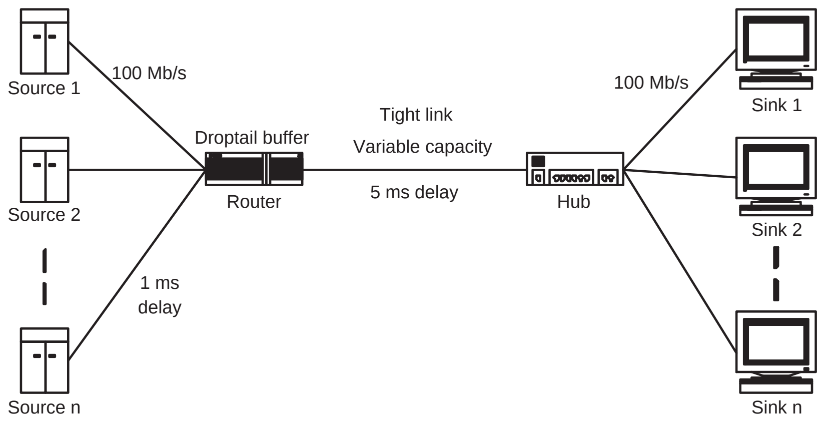
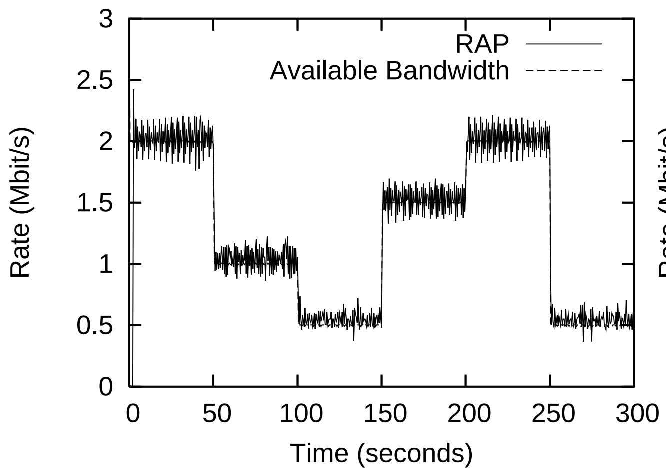
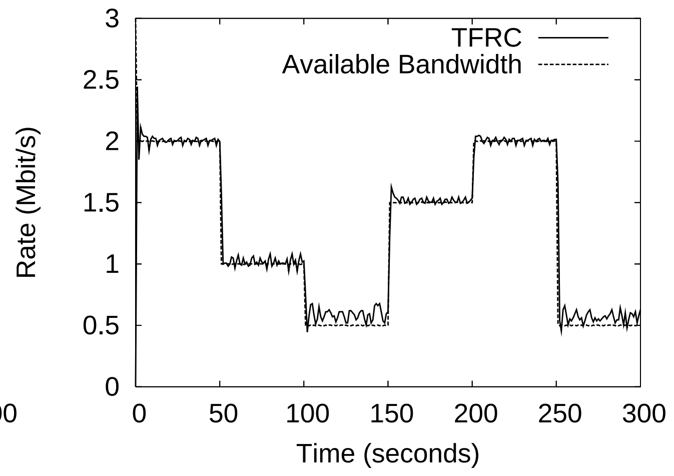


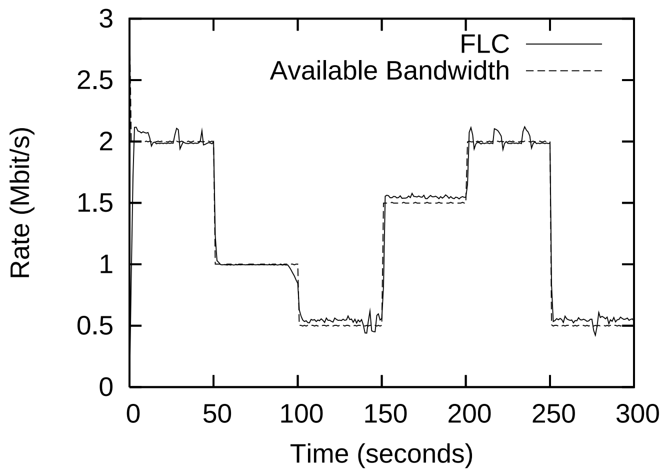
Figure 4: Dumbbell network configuration.



(a) RAP



(b) TFRC



(c) FLC

Figure 5: Tracking a changing available bandwidth showing RAP, TFRC and FLC sending rates.

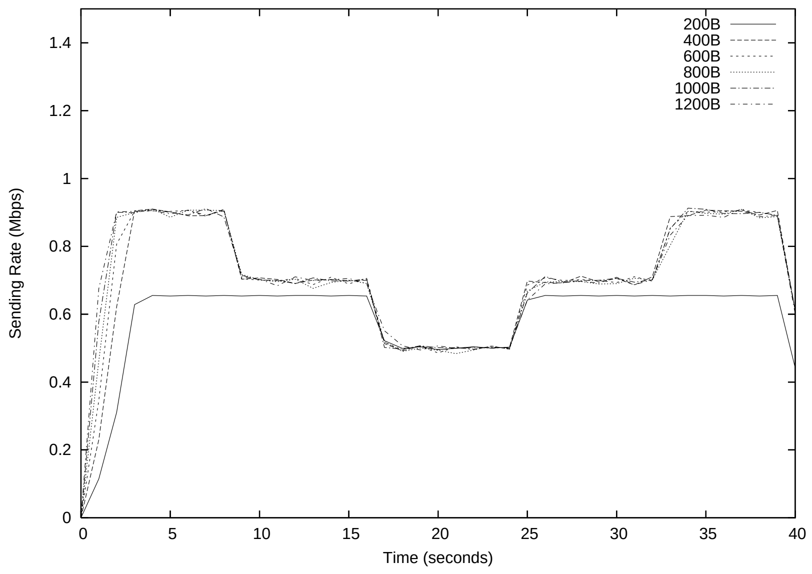


Figure 6: FLC instantaneous sending rates with different maximum packet sizes in the presence of step-wise changing available bandwidth.

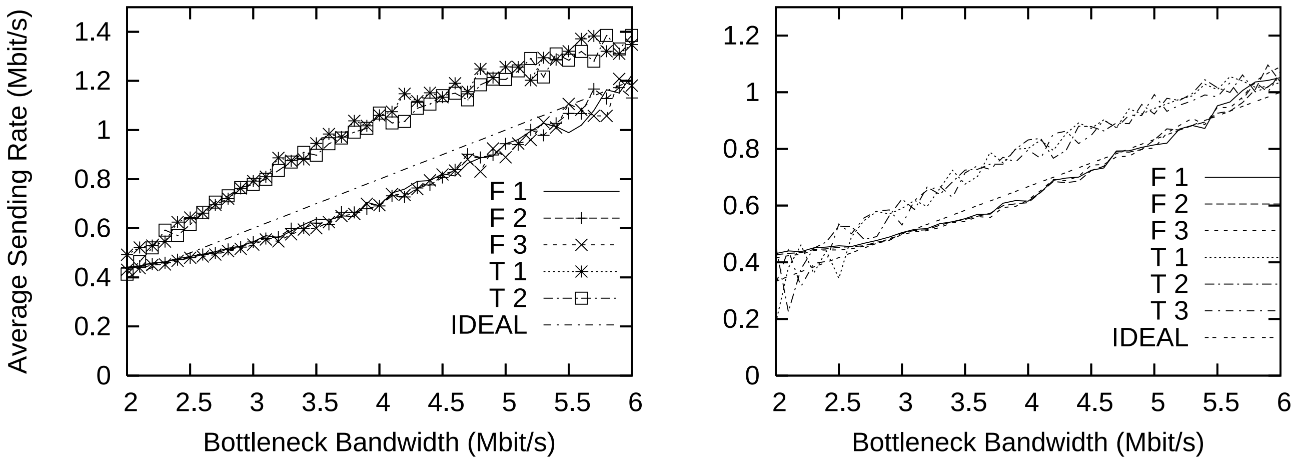


Figure 7: Multiple competing input flows with different controllers across a range of bottlenecks, compared to an ideal per-flow allocation of bandwidth.

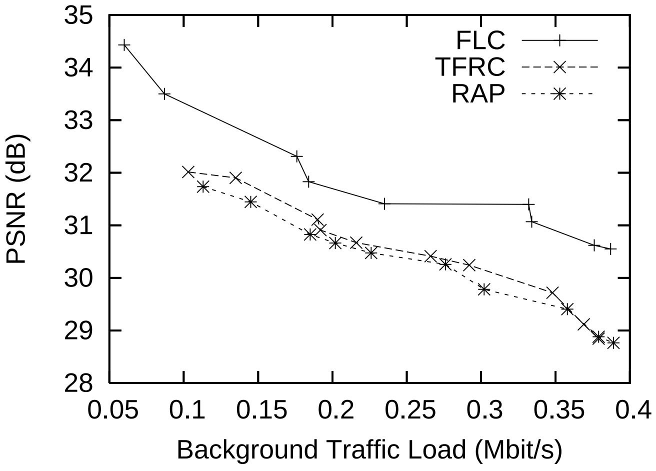


Figure 8: Received mean Y-PSNR of a VBR video stream (‘news clip’) with an increasing background traffic load for a 500 kb/s bottleneck bandwidth using either FLC, TFRC, or RAP.

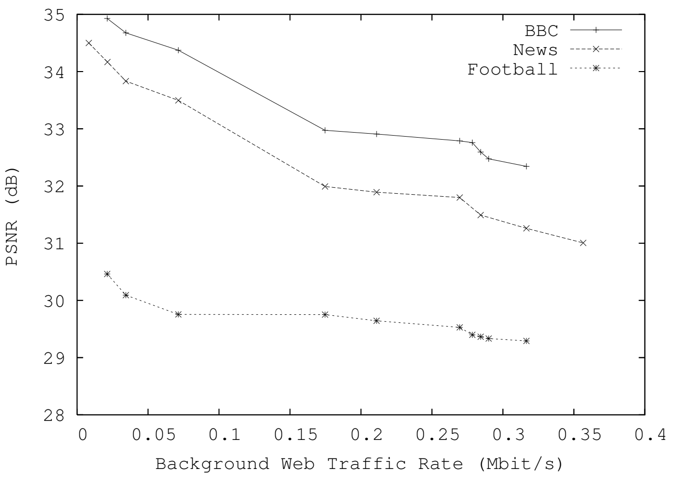
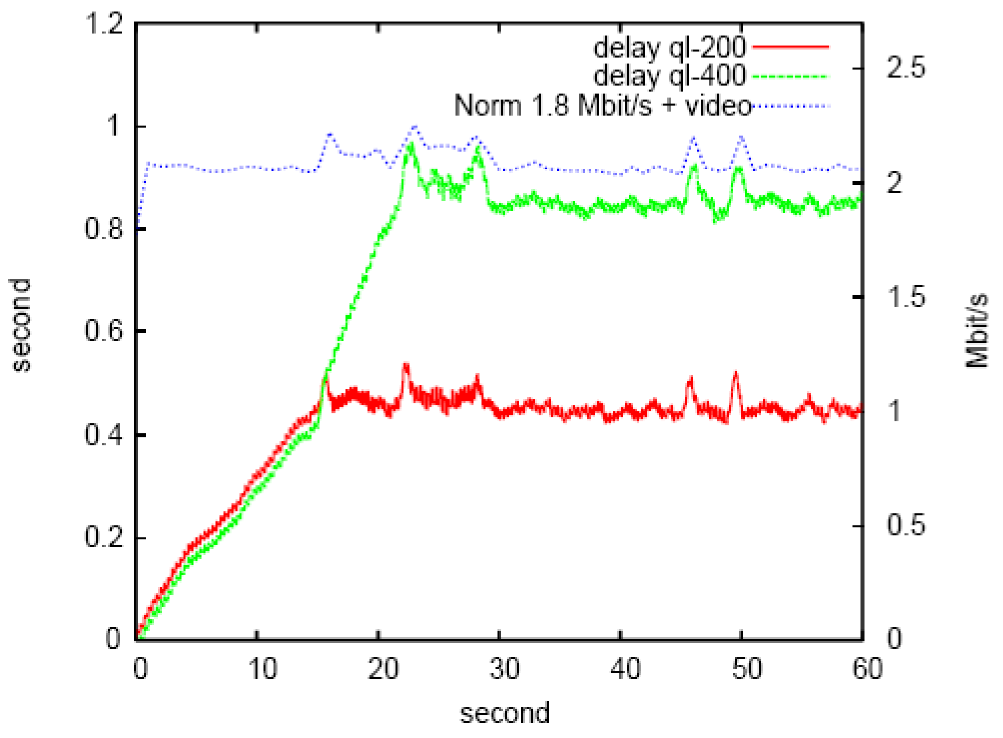
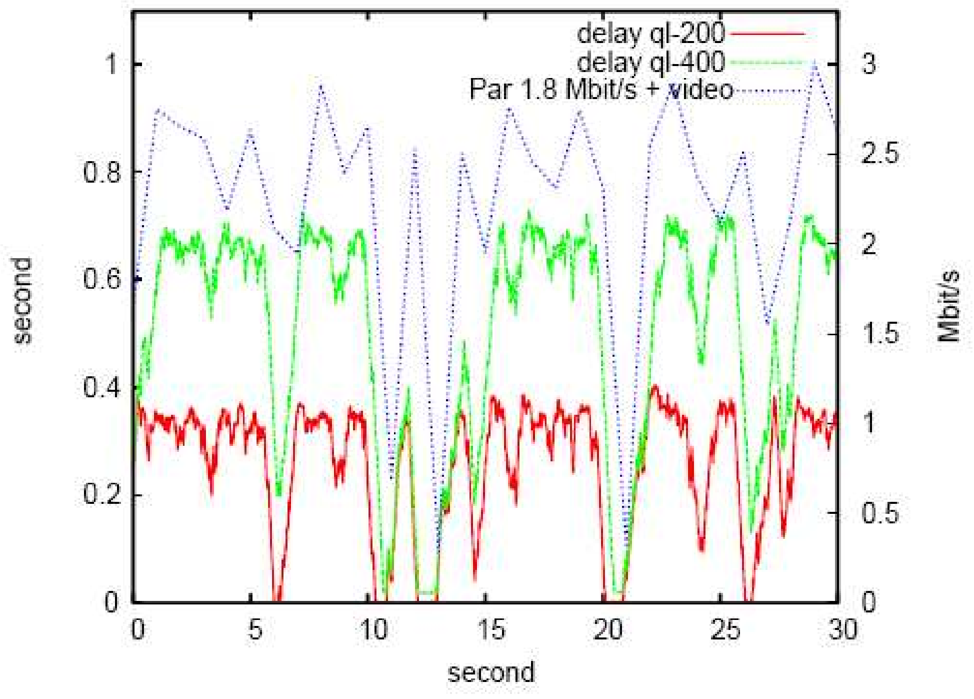


Figure 9: Received mean Y-PSNR of three VBR video stream with an increasing background traffic load for a 500 kb/s bottleneck bandwidth using FLC.



(a)



(b)

Figure 10: Sending rate and delays for two different packet queue lengths (Ql) with (a) Normal and (b) Pareto cross-traffic (1.8 Mbit/s).