

# Delay-based Congestion Avoidance for Video Communication with Fuzzy Logic Control

Emmanuel Jammeh, Martin Fleury, and Mohammed Ghanbari  
ESE Department, University of Essex, U.K.

**Abstract**— Early detection of network congestion is important in streaming video, as packet loss has a recurring impact on video quality. Packet delay itself, rather than packet loss, can give early notice of congestion provided it can accurately reflect the congestion level at the network path’s tight link. In this paper, the one-way delay of video packets serves as an incipient network congestion indicator, which acts as one input to fuzzy logic control of congestion avoidance to help optimize the response to network congestion. The fuzzy logic models are shown to be robust under: changes in the complexity and motion content of the video stream under control; a wide range one way end-to-end link delays up to 120 ms; and variations in available bandwidth. Tests reported that the fuzzy logic approach compares favorably to standard TFRC congestion control.

## I. INTRODUCTION

Networked video communication is achieved by determining the available bandwidth and adapting the video rate at a live-video encoder or an intermediate transcoder. It is paramount that the network state be determined in an accurate and timely manner. Packet loss has traditionally been used by TCP to signal congestion with remarkable success in avoiding excessive Internet congestion [1]. However, the limitation of packet loss as a congestion signal for high-bandwidth delay networks has also been identified as a performance bottleneck in TCP and many enhancements to the protocol involving a delay-based indicator have been proposed, for example [2][3][4]. For transport of encoded video, any reference picture packet loss is significant, as the effects endure until the next intra-coded picture. Nonetheless, TCP emulators [5] for video transport over UDP (to avoid unbounded delivery delay) do include a packet loss factor in their models.

In this paper, we consider whether delay-based congestion avoidance should be introduced to video transport, including networks with up to 120 ms of one-way latency, as may occur with trans-national connections. We also apply fuzzy logic control (FLC) to this problem, because the inherent looseness of its definition and the measurements available, together with the need for a real-time solution, all point in this direction. Unlike packet loss, which supplies one-bit of information on network congestion, *i.e.* whether the network is congested or not, packet delay provides multi-bit information on the level of congestion that can serve to give early notice of the onset of full-blown congestion.

Assume an application with input sending rate  $R_i$  and tight link (one with least available bandwidth on an end-to-end path) capacity  $C_b$ . For duration  $\delta t$ , the total

incoming traffic,  $T_i$ , into a tight link is given by

$$T_i = (R_i + R_c) \times \delta t \quad (1)$$

where  $R_c$  is the cross-traffic rate. The achievable outgoing rate across a bottleneck link during the same period is given by:

$$T_o = C_b \times \delta t. \quad (2)$$

Clearly, there will be no backlog of data on this link if the capacity is greater than the total incoming traffic. However, excess data will be buffered by the tight link router when the incoming rate is greater than the outgoing rate. This buffering results in a queuing delay,  $q_d$ , which is a function of the difference between the input and output rates of the tight link:

$$q_d = \frac{T_o - T_i}{C_b}. \quad (3)$$

The queuing delay  $q_d$ , which adds to the One-Way Delay (OWD) of a packet, is an indicator of network congestion. In Fig 1, once the rate  $R_i$  exceeds the available bandwidth then delay begins to climb. However,  $q_d$  cannot be measured directly, whereas a calculation of OWD can be made at a receiver. In practice, OWD cannot be measured accurately without clock synchronization, owing to the problem of clock drift. However, as finding  $q_d$  is the objective, approximate cancelation of the offset by subtraction of two OWDs is possible as further discussed in Section III-A. Minimum OWD in theory occurs when there is no queuing delay on the network path, with propagation delay being the only contributor. There will normally be some buffering delay included in the OWD calculation but  $q_d$  at a tight link will change as the link becomes congested. Increase in OWD is a function of an increase in buffer fullness, which in turn is a function of the level of network congestion. Maximum queuing delay occurs when the buffer at the tight link becomes full and packets start to be dropped: full-blown congestion. The queuing delay of packets with respect to the minimum calculated OWD is an indication of the tight link’s buffer fullness which we call multi-bit information, as it has a level which ranges between the minimum measured and maximum possible delay. To estimate congestion level from delay samples, we employ FLC.

FLC, which has from its inception [6] been extensively used for industrial and commercial control applications [7], is a convenient tool for handling un-modeled network

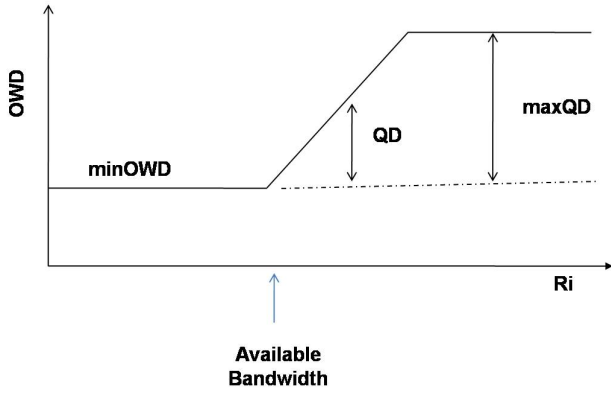


Fig. 1. Relationship between input rate and OWD

congestion states. It allows the intuitive nature of congestion reduction to be applied through linguistic variables [8]. Within video coding it has found an application [9][10] 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. In our application, FLC of congestion avoidance is a sender-based system for unicast flows. The receiver returns a feedback message that indicates averaged delay. This allows the sender to compute the network congestion level and from that the FLC estimates the response. We apply FLC in an unconventional fashion in that, while one input is fuzzy, another input is a binary test of the congestion trend (increasing or decreasing). Based on the FLC's estimate, the sender then applies a control signal to a real-time encoder or transcoder to alter the quantization level, as a reflection of the anticipated congestion. Rather than a single, invariant throughput equation to emulate TCP's behavior [11], FLC provides a flexible method of adapting to evolving Internet traffic patterns, as precise mathematical representations are difficult to devise for such a complex, non-linear control problem [12].

FLC is frequently efficiently implemented by means of a look-up table (LUT) of quantized model output values. For a hardware congestion controller, as advocated in [13], an LUT-based approach is a distinct advantage, though there are a range of designs for fuzzy logic hardware [14]. The need for a hardware implementation clearly implies that once developed the FLC models should have wide applicability without retuning. For example, the same controller should be able to cope with a range of Internet path delays and with video streams with differing characteristics in terms of scene complexity, motion, and scene cuts. This paper goes some way to verify that behavior.

The remainder of this paper is organized as follows. As the key to successful closed loop congestion control is the nature of the feedback information Section II examines previous delay-based mechanisms. Section III principally presents the FLC methodology, while Section IV details our simulation experiments to date. Finally, Section V

draws some conclusions.

## II. RELATED WORK

Several measurement schemes have investigated delay as a way of determining available network bandwidth. Direct probing [15] allows packet pair techniques [16] to be employed. Because direct probing assumes that the tight link is also the same as the link with minimum capacity (the thin link), this and other restrictions reduce its generality. Therefore, iterative probing with packet trains is to be preferred. Train of packet pairs (TOPP) [17] is an active probing scheme that injects multiple trains of packets pairs with differing intra-pair separations into the network under test. The sending rate of each train is linearly increased by reducing the inter-pair separations. The one-way packet pair dispersions are measured as an indication of delay experienced in queues along the network path. Pathload [18] applies periodic packet trains, which, it is said, overcomes a weakness of TOPP: that it is affected by the order of queues and their sizes. Pathload measures the trend of OWD for each packet within a long packet train to determine a self-induced loading effect, induced as the earlier part of a train occupies buffers. Pathload itself has long convergence times (10-30 s). PathChirp [19] reduces the convergence rate by varying the probing rate within each packet train. Both Pathload and pathChirp measure OWD but, as these methods induce congestion, they are not suitable for streaming applications. Other iterative probing tools include PTR [20] and Bfind [21].

As an example of probing directly applied to streaming, the Loss Delay-based Adjustment algorithm (LDA) [22] uses periodic pairs of video packets to estimate available bandwidth through packet dispersion at the tight link. However, LDA actively increases the video sending rate according to the available bandwidth until packet losses occur, rather than set the rate from the estimated delay.

The main intention of delay-based control is to avoid the need to rely on detrimental packet losses, and, therefore, delay-based control is a form of congestion avoidance [23][24]. In [25], delay-based congestion avoidance, using the delay gradient [26], was applied to interactive applications such as video conferencing to find the minimum possible delay without overly restricting throughput. In the process, it was found that output oscillations were reduced along with delay variance. The method also avoided the 'phase effect' [27], 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, owing to an ordering effect. Finally, one should add that recent measurements of packet loss and delay across a typical tight indicate that packet loss may be a weak indicator of congestion level because, when cross-traffic across the link is 'bursty', losses may not be evenly distributed between the competing flows.

There has been a recognition of the benefits of employing delay and delay variation as an early network

congestion indicator. Unfortunately, in Tahoe and New Reno TCP, Karn's algorithm implies that the round-trip time (RTT) of retransmitted packets (lost through congestion) cannot be used in RTT estimates. TCP Vegas [2] employs RTT rather than packet loss to indicate congestion. RTT is anyway an unreliable estimator of delay as it does not account for asymmetric paths, which occur through load imbalances. Therefore, TCP Santa Cruz [3] measures relative delay between packet pairs on their forward path. FAST TCP [4] finds changes in RTT to determine the onset and level of network congestion. Synch-TCP [28] is another proposed enhancement to TCP with early network congestion detection. The main difference between Synch-TCP and FAST TCP is that Synch-TCP measures delay from one-way transit time (GPS synchronized clocks to avoid clock drift may be necessary), while FAST TCP uses RTT instead. Unless firewalls are a problem for UDP transport, TCP is not normally considered for video transport, as without large playback buffers its reliability mechanism may result in accumulated packet delays. TCP's Additive Increase Multiplicative Decrease (AIMD) may also result in sending rate fluctuations.

A survey of congestion control through computational intelligence [29] observes that not much work has been reported on deploying natural algorithms within the Internet. The authors of [30] have explored fuzzy logic to improve the performance of the Random Early Discard (RED) Internet router queue algorithm, including Explicit Congestion Notification (ECN), and research in [31] considers DiffServ buffer occupancy for each class of layered video packets.

### III. CONGESTION DETECTION, AVOIDANCE, AND CONTROL

Fig. 2 shows the streaming architecture in which fuzzy logic controls the sending bit rate. The congestion level determination (CLD) unit finds the congestion state of the network from measured delay and delay variation made by the timer module. The congestion state data are relayed to the sender. FLC employs this multi-bit delay information to compute a new sending rate that is a reflection of the current sending rate and the level of network congestion. The video rate adaptation unit (either a transcoder adapting pre-encoded video or an encoder) changes the sending rate to that computed by the fuzzy controller. The current implementation changes the quantization level of a frequency-domain transcoder [33] but transmoding (changing the rate by altering the emphasis given to regions of interest or objects), or Fine-Grained Scalability are other possibilities.

#### A. Congestion Detection

The CLD module determines incipient congestion from queuing delay. For each received packet indexed by  $i$

$$OWD_i = T_r - T_s, \quad (4)$$

where  $T_r$  is the receive time of the current packet and  $T_s$  is the time the packet was sent (as found for instance

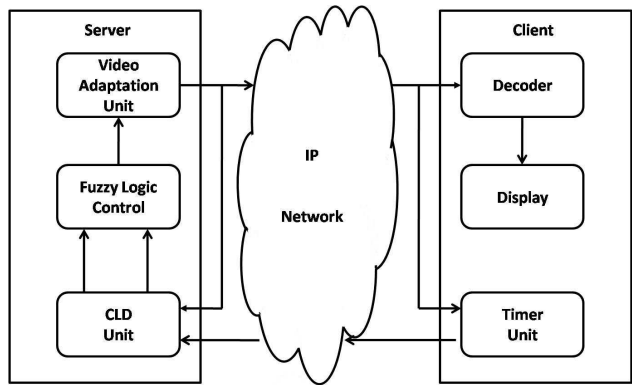


Fig. 2. Block diagram of video streaming architecture

from a Real-Time protocol (RTP) [32] packet header). When it is appropriate, the computed  $OWD_i$  updates the minimum and maximum one-way delays,  $OWD_{min}$  and  $OWD_{max}$ , on a packet-by-packet basis. Subsequently, the maximum queuing delay is found as  $maxQD = OW D_{max} - OW D_{min}$ . The queuing delay over the network path,  $QD_i$  is computed from the measured delay and the minimum delay:

$$QD_i = OW D_i - OW D_{min} \quad (5)$$

and an exponentially weighted average of the queuing delay for the  $i^{th}$  received packet is formed by,

$$avgQD_i = (1 - \alpha) \times avgQD_{i-1} + \alpha \times QD_i \quad (6)$$

where  $\alpha \leq 1$  is the forgetting constant. In simulations,  $\alpha$  was set to 0.1.

The queuing delay is a measure of network congestion, and the ratio of the average queuing delay to the maximum queuing delay is a measure of bottleneck link buffer fullness. A delay factor ( $d_f$ ) is computed from the average queuing delay and the maximum queuing delay,

$$d_f = \frac{avgQD_i}{maxQD} \quad (7)$$

where  $d_f$  ranges between [0,1] with 0 indicating no incipient congestion, 1 indicating full-blown congestion, with shades of incipient congestion between 0 and 1.  $d_f$  is an early notification of congestion.

It is difficult to determine the average queuing delay and whether it is increasing or not, given that background cross traffic can cause the queuing delay to fluctuate around the average. Methods are needed to determine whether the general trend of the average delay is increasing or not. Two trend analysis methods [18] determine whether the queuing delay is increasing or decreasing. In a given measurement epoch, a number  $k$  of queue delay samples are grouped into  $\tau$  groups where

$$\tau = \sqrt{k} \quad (8)$$

We use the pairwise comparison test (PCT) and the pairwise difference test (PDT) to determine the overall

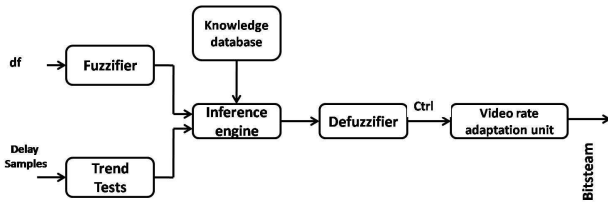


Fig. 3. Block diagram of FLC module

trend of the queueing delay, *i.e.* for PCT

$$T_{PCT} = \frac{\sum_{i=2}^{\tau} I(M^i > M^{i-1})}{\tau - 1}, \quad (9)$$

where  $M^i$  is the median of group  $i$  and  $I(X)$  is 1 if  $X$  holds and 0 otherwise, and for PDT

$$T_{PDT} = \frac{M^{\tau} - M^1}{\sum_{i=2}^{\tau} |M^i - M^{i-1}|}. \quad (10)$$

The trend is assumed to be increasing if  $T_{PCT} > 0.55$  or  $T_{PDT} > 0.44$ , with the threshold values established in tests reported in [18]. Otherwise, the trend is assumed to be decreasing.

If measurements of the  $OWDs$  in (5) are close in time, then the subtraction may remove the offset in time resulting from relative clock drift between the sender and the receiver. It is assumed that clocks are set by the Network Time Protocol (NTP), which provides sub-millisecond resolution on LANs and tens of millisecond accuracy on WANs. There are a number of algorithms for subsequently removing the effect of drift whether between NTP updates or otherwise. For example, work in [34] analyzes online skew correction through taking the convex hull of delay measurements. Hardware synchronization is also possible [28]. In (5), one of the  $OWDs$  is an estimate of the minimum  $OWD$ . Owing to dynamic packet routing, it is possible that the  $OWD_{min}$  will vary over time, as the network path changes. Therefore, another possibility is sampling of the  $OWD_{min}$  within a time localized window. This has the by-product that estimates of  $OWD_{min}$  are made closer in time to the current estimate of  $OWD$ .

### B. Fuzzy Congestion Avoidance and Control

Fig. 3 is a block diagram of the FLC module. 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. 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 inferred fuzzy control decisions from the inference engine to a crisp or precise value, which is converted to a control signal.

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 ... 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. For example, Matlab’s toolbox allows a set of output data points to be calculated to a given resolution, allowing interpolation of the surface.

Possible input,  $d_f$ , and output variables for this FLC are summarized in Table I. Notice that though these linguistic variables are identified in Table I, this does not imply that all of them are utilized in the implementation of the fuzzy membership functions, and indeed some were found to be extraneous to the needs of the fuzzy models in the following. The congestion trend is not fuzzified but is either increasing (1) or decreasing (0). The input variables were fuzzified by means of triangular-shaped membership functions, being a 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. Fig. 4 describes for  $d_f$  its membership functions that partition its range into four regions with different levels according to Table I. Similarly, Fig. 5 shows the partitioning of the output, which again is listed in Table I.

Dividing the congestion levels into a manageable set ranging from low to very high (according to delay) makes

TABLE I  
LINGUISTIC VARIABLES FOR FLC INPUT AND RESULTING OUTPUT

$d_f$		output	
<i>L</i>	Low	<i>NL</i>	Negative Low
<i>M</i>	Medium	<i>NM</i>	Negative Medium
<i>H</i>	High	<i>NH</i>	Negative High
<i>VH</i>	Very High	<i>NVH</i>	Negative Very High
		<i>NEH</i>	Negative Extremely High
		<i>Z</i>	Zero
		<i>PL</i>	Positive Low
		<i>PM</i>	Positive Medium
		<i>PH</i>	Positive High
		<i>PVH</i>	Positive Very High
		<i>PEH</i>	Positive Extremely High

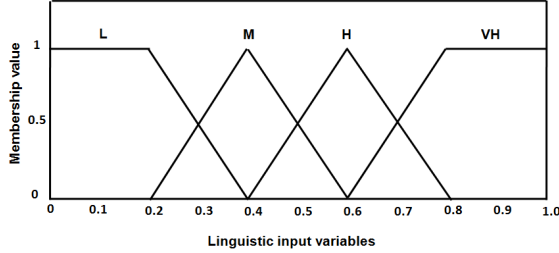


Fig. 4. Delay factor membership functions

for a simple number of decision rules, as expressed in Table II. The fuzzy inference rules are in the same form as the following examples:

if  $d_f$  is H and T is I then S is NVH  
if  $d_f$  is L and T is D then S is PVH

where S is the fuzzified output from the controller, D is a decreasing trend (T) and I is an increasing trend, while the output is taken from Table I. The complete set of rules for the evaluation of the control output in Table II capture in a concise form the information contained in English sentences constrained in the manner of the previous two examples. The FLC employs a simple Mamdani inference model [35] and center-of-gravity defuzzification method. Eqn. (11) maps the input to the output of the controller:

$$Ctrl = \frac{\sum_{i=1}^M S_i K_i}{\sum_{i=1}^M K_i} \quad (11)$$

where  $M$  is the number of rules,  $S_i$  is the value of the output for rule  $i$ ,  $K_i$  is the inferred weight of the  $i^{th}$  output membership function. More specifically,  $S_i$  is the value at the middle of the range of data values that are possible

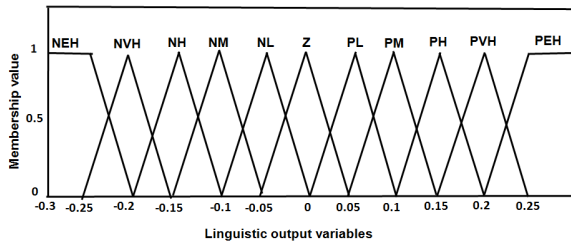


Fig. 5. Output membership functions

members of the  $i^{th}$  fuzzy subset.  $K_i$  is the area under the  $i^{th}$  output membership function, clipped by the minimum possibility of membership of  $d_f$  in the input membership function of the  $i^{th}$  rule. As more and more rules may apply, (11) 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. In a future implementation, it is possible to fuzzify the trend variable between the two current Boolean inputs of 0 and 1. Fig. 6 includes this possibility, whereas, at present, the output is plotted across the facing and rear 2D  $d_f$  vertical planes of this surface.

TABLE II  
TABLE SHOWING THE FUZZY-LOGIC INFERENCE RULES

<i>Trend</i>	<i>Delay, d<sub>f</sub></i>			
	<i>L</i>	<i>M</i>	<i>H</i>	<i>VH</i>
<i>D</i>	PVH	PM	Z	NL
<i>I</i>	PM	NL	NH	NEH

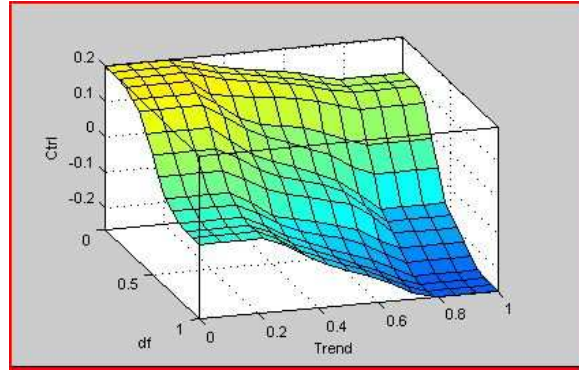


Fig. 6. Output control surface (with fuzzy 'trend' input)

The control signal  $Ctrl$ , as specified in (11), is normalized to the range (0, 1], subject to a minimum lower bound (to avoid unacceptable low quality video output). For input bitrate  $R_{in}$ , the target output bitrate is  $R_{out}$  is given by,

$$R_{out} = (1 + Ctrl) \times R_{in}. \quad (12)$$

#### IV. EXPERIMENTS

The algorithm was simulated with the well-known ns-2 network simulator (version 2.28 used). The simulated network, with a typical dumbbell topology, 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 initially set to 5 ms and the side links delays were set to 1 ms. The tight links queueing policy was defaulted to be 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.

##### A. Feedback

Network-state decisions were fed back from the receiver to the fuzzy controller after at least 40 ms had

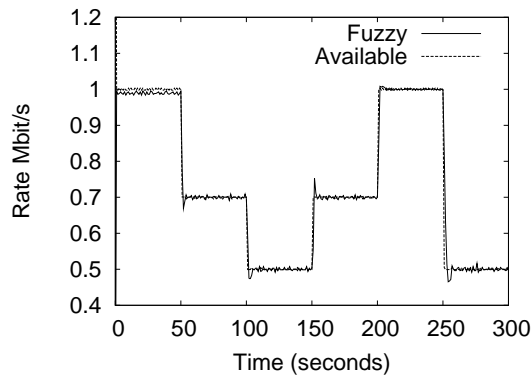


Fig. 7. Tracking a varying available bandwidth

elapsed at the receiver, which corresponds to a rate of 25 frame/s. As becomes evident in the reported tests, for larger network path delays (at a transcontinental scale), more frequent feedback messages would be needed to reduce the overall latency, improving the response. Increasing the sampling rate was also advocated for TCP RTT measurements in RFC 1323. Some consideration to the stability of the control system might also be necessary in those circumstances. However, for expected streaming services within medium-sized networks, these measures are unlikely to be necessary. For example, in the congestion control system of [36] latencies beyond 120 ms were not tested and stability was apparently not an issue, which is the same assumption in this paper.

For reliability, a TCP feedback channel connection is preferred. However, in the simulations, as at most (for Web background traffic) only ACK packets were present on the feedback channel, UDP was the transport protocol. In the simulations, the arrival of a UDP feedback packet triggers a sending rate decision calculation. Reporting on issues such as the impact of cross traffic on feedback and choice of TCP over UDP for feedback messages is reserved for a further paper.

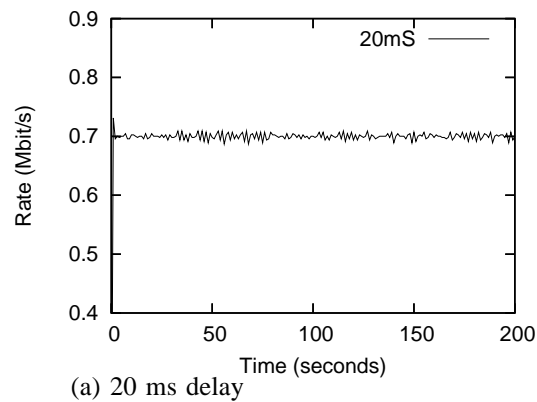
In an implementation, for unicast streaming, it is envisaged that feedback messaging over Berkeley sockets would be a sufficient and not require RTP's companion protocol, Real-time Control Protocol (RTCP) (RFC 3550). For large scale, possibly multicast extensions [37] then RTCP should be contemplated.

### B. Tracking Bandwidth

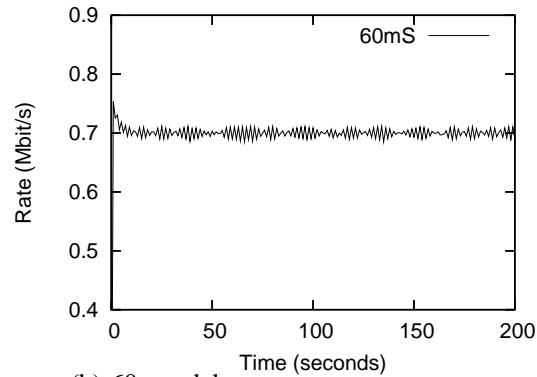
Calibration experiments indicated that the algorithm accurately tracked various fixed capacity bottleneck bandwidths, producing a smooth flow with limited fluctuations to disturb delivered video quality. The performance of the scheme was also tested by changing the available bandwidth during the streaming session by injecting Constant Bit Rate (CBR) background traffic at various rates. In Fig. 7 it is clear that the algorithm responds to the changes in a timely manner, after a brief period of adjustment.

### C. Varying End-to-end Delay

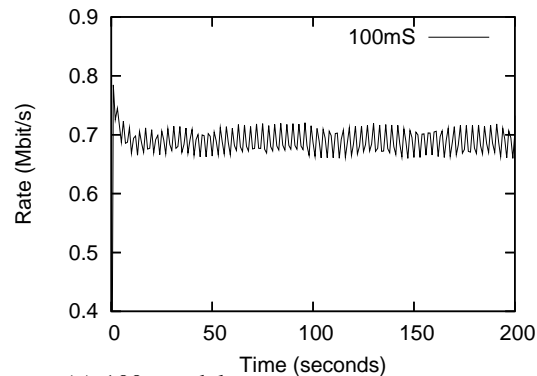
The performance of the algorithm was tested by varying the tight link one-way latency in order to emulate a range



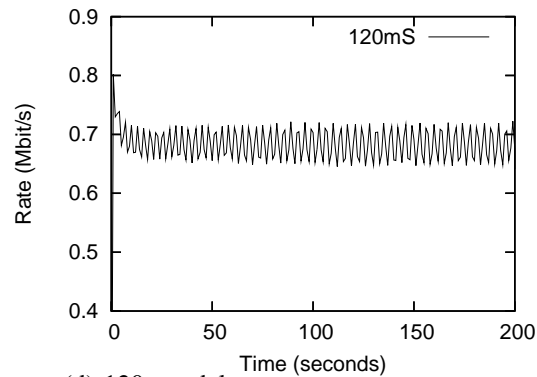
(a) 20 ms delay



(b) 60 ms delay

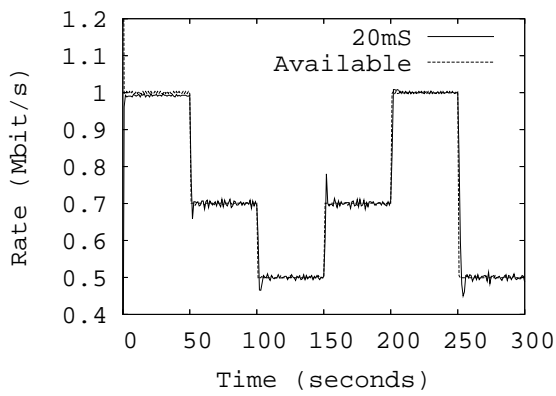


(c) 100 ms delay

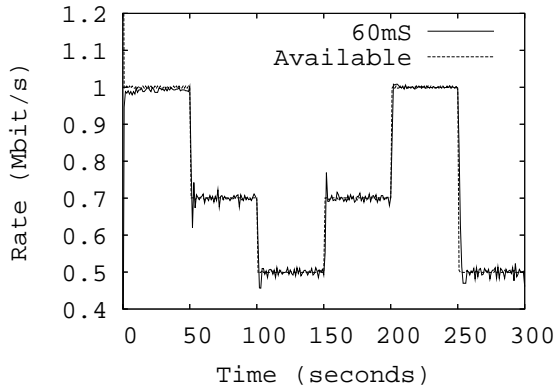


(d) 120 ms delay

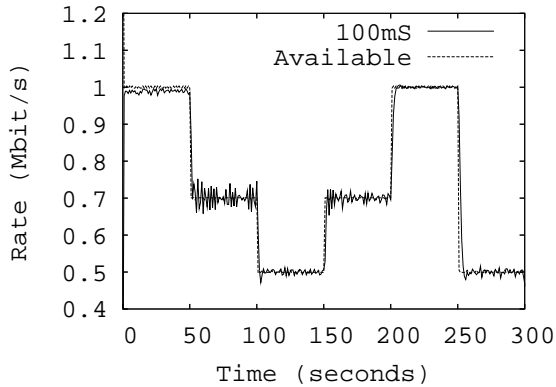
Fig. 8. FLC rates across a 700 kbit/s bottleneck with various end-to-end delays



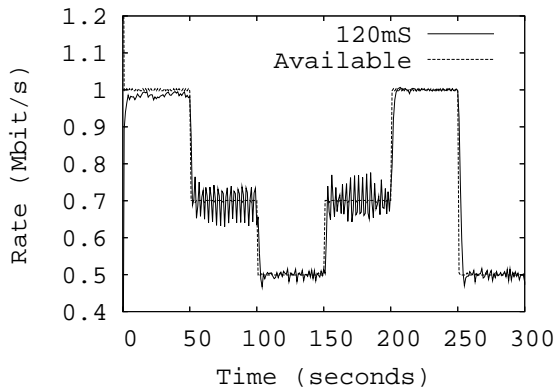
(a) 20 ms delay



(b) 60 ms delay



(c) 100 ms delay



(d) 120 ms delay

Fig. 9. FLC rate tracking a varying available bandwidth as end-to-end delay increases

of end-to-end path delays. Whether the delay arises from propagation or from queuing delay, the feedback to the FLC module will be delayed. The tight link capacity was set to 700 kbit/s in Fig. 8. The FLC CBR rate maintains its value, though the sending rate becomes more ‘bursty’ for delays of 100 ms and above, as feedback is itself delayed. Therefore, there is a possibility of an unsatisfactory experience for the viewer as the received quality changes for path delays at and beyond 120 ms. However, all congestion control mechanisms will suffer from the same problem and can only reduce the problem by increasing the feedback sampling rate. In our case, this would gain at most a 40 ms reprieve, at a cost in reduced number of samples to make the delay estimate, which itself may lead to rate oscillations. As mentioned in Section I, there may be few situations when delays of over 120 ms occur in streamed video applications.

The algorithm was further tested by varying both available bandwidth and end-to-end path delay. Fig. 9 reports the result for the same range of delays as in Fig. 8, with similar conclusions. As the feedback path becomes longer in duration, the FLC rate begins to drop below the available bandwidth. However, this is to be preferred, as more excursions above the available bandwidth lead to packet loss.

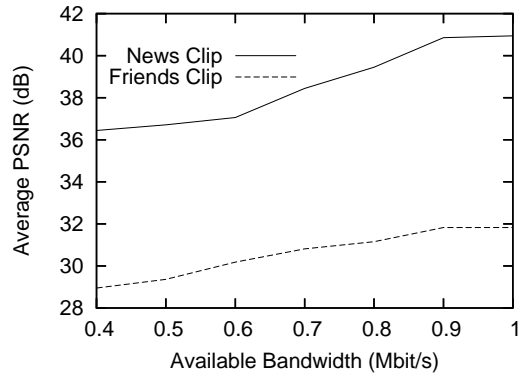


Fig. 10. Average PSNR for different available bandwidths when sending under FLC for ‘News’ and ‘Friends’ sequences

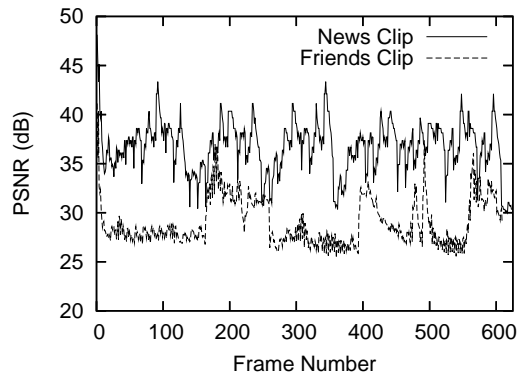


Fig. 11. PSNR for a 400 kbps available bandwidth when sending under FLC for ‘News’ and ‘Friends’ sequences

#### D. Video quality

The average PSNR was found for two 32 s video clips: 1) ‘News’, consisting of a news presenter and changing background, with moderate motion; and 2) an extract from an episode of the well-known situational comedy ‘Friends’, with greater motion. The videos were encoded using MPEG-2 at an original bit-rate (before transcoding) of 1 Mbps. European SIF-format sequences were used (progressive, 25 frame/s,  $352 \times 288$  pixels). The pictures were divided into eighteen per row of macroblock slices, with one slice in each packet for error resilience purposes.

Fig. 10 shows average PSNR after transmitting these clips across a link with differing available bandwidth (formed from a CBR background flow), with delay set 5 ms. Clearly, the video quality tracks available bandwidth but the quality is reduced for the more active ‘Friends’ clip. Given that PSNR is a logarithmic quantity, the bottleneck tracking is not expected to be strictly linear and, at lower bandwidths, the transcoder implementation limited the rate to around 90% of the input rate. Fig. 11 shows the time-wise behavior of PSNR for the two clips under the *worst* of the bottlenecks. Without the impact of cross traffic, the quality, relative to the original encoded video, is reasonable, though clearly the more complex ‘Friends’ clip’s quality has been degraded more.

The ability to track a changing available bandwidth is illustrated by Fig. 12. This is the same test as in the previous Section but with the ‘News’ clip rather than CBR traffic under FLC. Also shown is the *Ctrl* output from the FLC module, plotted against the left-hand vertical axis. Again, apart from small over-rides at the time of an abrupt upward available bandwidth change and some flattening out of the rate after a downward change, tracking accuracy is maintained. In practice, available bandwidth would not change as abruptly as happens in this test. Optimization of the fuzzy models will further adjust these small inconsistencies.

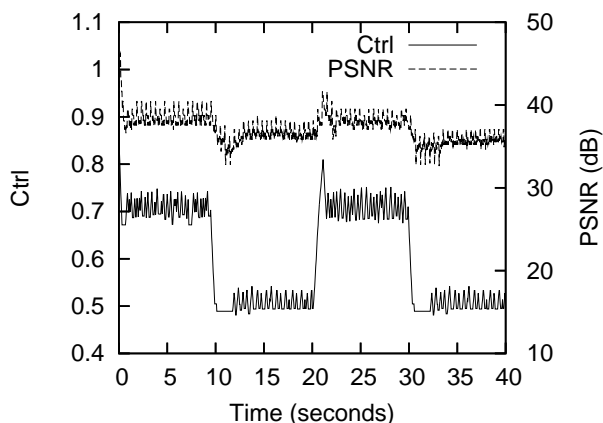


Fig. 12. FLC of the ‘News’ sequence with changing available bandwidth and including the *Ctrl* output from the FLC.

A set of tests were also performed to establish whether available bandwidth tracking was maintained in typical Internet conditions. Internet measurement studies [38][39]

have demonstrated a typical Internet traffic mix to consist of longer term flows, ‘Tortoise’, representing file transfers, and transient HTTP connections, ‘Dragonflies’. In our set of tests, one FLC video source and ten TCP sources were passed across the link. The first 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. The remaining 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.

Ten experiments were conducted for a bottleneck capacity of 1 Mbit/s, with a 5 ms delay across the link. In the first experiment, only one TCP ‘tortoise’ source was present as background traffic, in the second two TCP ‘tortoise’ sources acted as background traffic and so on until the ‘dragonfly’ sources are eventually introduced, so that all ten TCP sources were on as background traffic for the tenth experiment. The video source was ‘News’ with the same characteristics as before. Fig. 13 summarizes the results, showing the correspondence between the mean over time of the bandwidth occupied by the FLC video source (plotted against the left-hand vertical axis) and the mean available bandwidth. The data points are not evenly spread out because of the larger bandwidth contribution of the ‘tortoise’ sources compared to ‘dragonflies’. Some data points, at low available bandwidth, are partially superimposed in the plot. For convergence, the data points represent the mean of twenty independent runs.

From the plot, the sending rate of the FLC video is never above the available bandwidth, but tracks the available bandwidth throughout the experiments, leading to low packet losses. Consequently, the video quality (mean PSNR over time plotted on the right-hand vertical axis) follows the rate trend and does not suffer any degradation in quality.

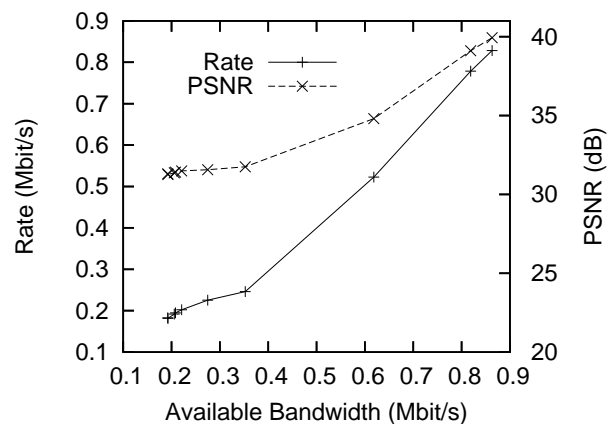


Fig. 13. FLC of the ‘News’ sequence with rising available bandwidth due to the presence of Web-like traffic, showing the mean bandwidth acquired by the FCL source and its corresponding video quality.

### E. Strength of the FLC solution

A comparison was made between the video quality delivered by FLC congestion avoidance and that by TCP Friendly Rate Control (TFRC) protocol, which is a prominent method of congestion control, now the subject of an RFC [40]. To ensure fairness the publicly available TFRC ns-2 simulator model (in the form of object tcl scripts to drive the simulator) was availed of from <http://www.icir.org/tfrc/>. 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].

In Fig. 14 and 15 both methods of congestion control were applied to the 'News' and 'Friends' sequences respectively. The available bandwidth at their various rates was formed by sending a CBR stream across the bottleneck link. Averages were taken over ten simulation runs. The simulation configuration was the default for this Section. Though the advantage is not as much for FLC when controlling the 'Friends' sequence than for 'News', for all data points the delivered video quality from FLC congestion avoidance is superior to that of TFRC's. Though this is not our main purpose, there are a variety of reasons that one might suggest for the weaker performance of TFRC such as: the retention of a residual bandwidth probing action out of the need to emulate the average rate of TCP; the fact that rate decisions are taken every RTT; and use of the RTT rather than OWD as input.

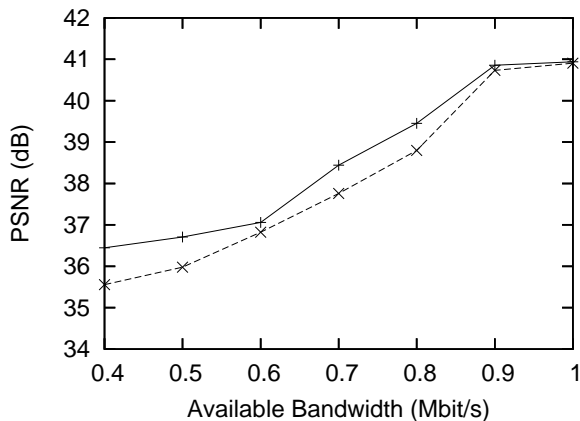


Fig. 14. FLC and TFRC: Comparison of average PSNR for a range of available bandwidths, when transporting the 'News' clip.

## V. CONCLUSIONS

This paper's contributions lie in two directions: 1) computational intelligence is applied to congestion detection rather than through a throughput modeling approach; and 2) delay acts as a congestion indicator without reliance on packet loss. Fuzzy logic control based on end-to-end delay information was able to track available bandwidth in a timely fashion. An unconventional design in which only one input was fuzzified, while the other tested the

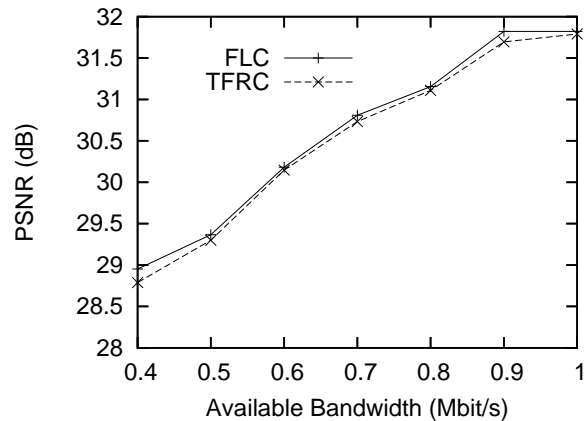


Fig. 15. FLC and TFRC: Comparison of average PSNR for a range of available bandwidths, when transporting the 'Friends' clip.

congestion trend was shown to be feasible. Currently, the method has proved to be robust in respect to changing delays and to differing types of input video clip. Even with delays of up to 120 ms the control method was still reasonably effective. Video quality was maintained even under a drastic restriction in available bandwidth, with some variation depending on the clip complexity. Fuzzy logic control also compares favorably with the well-known TFRC method of congestion control. The implication is that a stable set of fuzzy models can form part of a hardware congestion controller, which could work for a large range of network conditions. Future development of fuzzy logic for congestion control should consist of experiments on a larger testbed and/or across the live Internet. This development work should extensively explore the effect of varying the control parameters within the algorithm.

## ACKNOWLEDGMENT

This work was supported by the EPSRC, UK under grant no. EP/C538692/1.

## REFERENCES

- [1] M. Allman, V. Paxson, and W. R. Stevens, "TCP congestion control," *RFC 2581*, April 1999.
- [2] L. Brakmo, W. W. O'Malley, and L. L. Petersen, "TCP Vegas: New techniques for congestion detection and avoidance", *Proceedings of ACM SIGCOMM*, pp. 24–35, Aug. 1994.
- [3] C. Parsa and J. J. Garcia-Luna-Aceves, "Improving TCP congestion control over internets with heterogeneous transmission media", *7<sup>th</sup> IEEE Int. Conf. on Network Protocols*, pp. 213–221, Oct. 1999.
- [4] D. X. Wei, C. Jin, S. H. Low, and S. Hedge, "FAST TCP: Motivation, architecture, algorithm and performance", *IEEE/ACM Trans. on Networking*, vol. 4, no. 6, pp. 1246–1259, 2006.
- [5] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control", *IEEE Network*, vol. 15, no. 3, pp. 28–37, 2001.
- [6] L. A. Zadeh, "Fuzzy sets", *Information and Control*, vol. 8, pp. 338–353, 1965.
- [7] 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.
- [8] C. C. Lee, "Fuzzy logic control systems: Fuzzy logic controller – part 1", *IEEE Trans. Sys. Man and Cybern.*, vol. 20, no. 2, pp. 404–418, 1990.

- [9] A. Leone, A. Bellini, and R. Guerrieri, "An H.261 fuzzy-controlled coder for videophone sequences", *IEEE World Conf. on Computational Intelligence*, pp. 244–248, June 1994.
- [10] P. M. Grant, Y.-S. Saw, J. M. Hannah, "Fuzzy rule based MPEG video rate prediction and control", *Eurasip ECASP Conf.*, pp. 211–214, June 1997.
- [11] J. Padyhe, V. Firoiu, D. Towsley, and J. Krusoe, "Modeling TCP throughput: A simple model and its empirical validation", *Proceedings of ACM SIGCOMM*, pp. 303–314, Sept. 1998.
- [12] C. E. Rohrs, R. A. Berry, and S. J. O'Halek, "A control engineer's look at ATM congestion avoidance", *IEEE GLOBECOM*, vol. 2, pp. 1089–1094, Nov. 1995.
- [13] R. Rejaie, "On integration of congestion control with Internet streaming applications", *13<sup>th</sup> Int. Packet Video Workshop*, April 2003.
- [14] I. Baturone, A. Barriga, S. Sánchez-Solano, C. Jiménez, and C. López, *Microelectronic Design of Fuzzy Logic-based Systems*, CRC Press, Baton Rouge, FO, 2000.
- [15] M. Jain and C. Dovrolis, "Ten fallacies and pitfalls in end-to-end available bandwidth estimation", *4th ACM/USENIX Internet Measurement Conference*, pp. 272–277, Oct. 2005.
- [16] R. S. Prasad, M. Murray, C. Dovrolis, and K. Claffy, "Bandwidth estimation: Metrics, measurement techniques, and tools", *IEEE Network*, vol. 17, no. 6, pp. 27–35, Nov. 2003.
- [17] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks", *Global Internet Symposium*, pp. 415–420, Nov. 2000.
- [18] M. Jain and C. Dovrolis, "Pathload: A Measurement Tool for End-to-End Available Bandwidth", *Passive and Active Measurements Workshop*, March 2002.
- [19] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths", *Passive and Active Measurements Workshop*, April 2003.
- [20] N. Hu and P. Steenkiste, "Evaluation and characterization of available bandwidth probing techniques", *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 6, pp. 879–894, Aug. 2003.
- [21] A. Akella, S. Seshan, and A. Shaikh, "An empirical evaluation of wide-area Internet bottlenecks", *2nd ACM/USENIX Internet Measurement Conference*, 2003.
- [22] D. Sisalem and H. Schulzrinne, "The loss-delay based adjustment algorithm: a TCP-friendly adaptation scheme", *Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, pp. 215–226, Jul. 1998.
- [23] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks", *Computer Communications Review*, vol. 19, no. 5, pp. 56–71, 1989.
- [24] C. Jin, D. X. Wei, and S. H. Low, "The case for delay-based congestion control", *IEEE Annual Workshop on Computer Communications*, pp. 99–104, 2003.
- [25] W. Dabbous, "Analysis of delay based congestion avoidance algorithm", *High-Performance Networking IV*, pp. 283–298, 1992.
- [26] Z. Wang and J. Crowcroft, "Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm", *Computer Communications Review*, vol. 22, no. 2, pp. 9–16, 1992.
- [27] S. Floyd and V. Jacobsen, "Traffic phase effects in packet-switched gateways", *Computer Communications Review*, vol. 22, no. 2, pp. 26–42, 1991.
- [28] M.C. Weigle, K. Jeffay, and F.D. Smith, "Delay-based early congestion detection and adaptation in TCP: Impact on Web performance", *Computer Communications*, vol. 28, no. 8, pp. 837–850, May 2005.
- [29] A. Pitsillides and A. Sekercioglu, "Congestion control", *Computational Intelligence in Telecommunications Networks*, W. Pedrycz and A. Vasiliakos, Eds. CRC Press, Boca Raton, FL, Sept. 2000, pp. 109–158.
- [30] L. Rossides, C. Chrysostemou, A. Pitsillides, and A. Sekercioglu, "Overview of Fuzzy-RED in Diff-Serv networks", *Soft-Ware 2002*, pp. 2–14, LNCS # 2311, 2002.
- [31] X. Wang, D. Ye, and Q. Wu, "Using fuzzy logic controller to implement scalable quality adaptation for stored video in DiffServ networks", *12<sup>th</sup> Int. Packet Video Workshop*, April 2002.
- [32] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "RTP: A transport protocol for real-time Applications", RFC 3550, July 2003.
- [33] P. A. A. Assunção and M. Ghanbari, "Buffer analysis and control in CBR video transcoding", *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 83–92, 2000.
- [34] L. Zhang, Z. Liu, and C. H. Xia, "Clock synchronization algorithms for network measurements", *IEEE INFOCOM*, pp. 160–169, 2002.
- [35] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller", *Int. J. of Man-Machine Studies*, vol. 7, no. 1, pp. 1–13, 1975.
- [36] 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 Trans. on Circuits and Systems for Video Technology*, vol. 14, no. 2, pp. 256–268, Feb. 2004.
- [37] V. Novotny and D. Komosny, "Optimization of large-scale RTCP feedback reporting in fixed and mobile networks", *3<sup>rd</sup> Int. Conf. on Wireless and Mobile Communications*, pp. 85–91, 2007.
- [38] N. Brownlee and K. C. Claffy, "Understanding Internet traffic streams: Dragonflies and tortoises", *IEEE Communications*, vol. 40, no. 10, pp. 110–117, 2002.
- [39] J. S. Marron, F. Hernandez-Campos, and F. D. Smith, "Mice and elephants visualization of Internet traffic", *15<sup>th</sup> Conference on Computational Statistics*, Aug. 2002.
- [40] M. Handley, S. Floyd, J. Padyhe, and J. Widmer, "TCP friendly rate control (TFRC): Protocol specification", 2003, IETF RFC 3448.