Video Streaming in Evolving Networks under Fuzzy Logic Control

Martin Fleury, Emmanuel Jammeh, Rouzbeh Razavi,
Sandro Moiron and Mohammed Ghanbari
University of Essex
United Kingdom

1. Introduction

Internet Protocol TV (IPTV) and other video streaming services are expected to dominate the bandwidth capacity of evolving telecommunications networks. In fact, managed, all-IP networks are under construction with video largely in mind. In these networks, a variety of broadband access networks will form the final link to the home across which video is streamed from proprietary servers. Co-existing with these networks or as an extension of them, the traditional, best-effort Internet will continue to support applications such as video-on-demand, Peer-to-Peer (P2P) streaming, and video clip selection.
This Chapter will begin by broadly surveying research and development of video streaming across evolving telecommunications networks under the categories of best-effort and managed networks. In particular, the Chapter will introduce the different forms of control that are necessary to ensure the quality of the delivered video, whether live or pre-encoded video, when for the latter bitrate transcoding may be required. The concentration will be on single-layer unicast distribution though simulcast, bandwidth reservation, multicast, and other forms of delivery will be touched upon.
With the growth in computational power, rate-distortion (R-D) control has emerged as an effective way to optimise the output encoded bitstream. In R-D control, the optimal choice of compression rate (and hence codec output bitstream rate) relative to improvement in video quality is sought. (The choice is generally found through the method of pre-set Lagrangian multipliers with trial codec settings repeatedly tested to find the best result.) Though attempts have been made to integrate R-D control and network congestion control (Chou & Miao, 2006), often congestion control has been considered separately as best-effort networks are prone to fluctuations in available bandwidth. In all-IP networks, though traffic in the core of the network will be switched, the variety of access network types poses a problem to servers that may be oblivious of the final hop technology. When broadband wireless (IEEE 802.16 d,e, WiMAX (Fleury et al., 2009)) access links are involved error control is especially important.
The Chapter will then specialise to consider in what ways fuzzy logic controllers (FLCs) have been applied to rate control and congestion control. A feature of this Chapter will be consideration given to the growing prominence of type-2 fuzzy logic in networked

multimedia control, bringing greater robustness in the face of unforeseen network conditions. To illustrate the application of fuzzy logic control, the Chapter will include two case studies. One of these will show how type-2 logic can improve upon type-1 logic, both of which forms of congestion control improve upon traditional controllers respectively within managed networks and within the Internet. The design of the controllers is illustrated for non-specialists, showing how type-2 controllers extend type-1 FLCs. From the results of simulations, FLCs in a managed network are shown to be superior to traditional congestion controllers. Transcoding is presented as an effective way to apply fuzzy logic control.
With the advent of IPTV, statistical multiplexing has again become an important issue for managed networks. Unlike traditional broadcast channels, network distribution may involve changes in available bandwidth and streaming conditions because of the variety of possible access types and coexisting traffic. In the second case study, an FLC is used to integrate two video complexity measures to achieve an effective combination of video or TV channels. The intention is dynamically to reduce the bandwidth allocation to channels that are already of high enough quality and increase the quality of streams with potentially greater coding complexity. Simulation results are presented to show the value of the approach applying the state-of-the-art H.264 codec. This case study will also include a review of other forms of statistical multiplexing.

2. Video streaming

2.1 Streaming basics

In video streaming, the compressed video bitstream is transmitted across a network to the end user’s decoder (prior to display) without the need for storage other than in temporary buffering. Its advantage over progressive download from a network point-of-view is that the throughput is only that required to render the video at the user’s display. Download risks overloading the network by too high a throughput. If download is not progressive, then the user has to wait an intolerable time before (say) viewing a 2 hr movie. There are also issues of commercial confidentiality if the video is stored on the user’s machine.
Downloading video does permit Variable Bitrate (VBR) to be transported. In VBR, the codec quantization parameter (QP) is fixed leading to a constant quality. The alternative is to set a target bit rate for Constant Bitrate (CBR) video and allow fluctuating quality but with a gain in controllability. The main problem with VBR is that due to a strong variation in the number of bits allocated to each of the frame types (Lakshman et al., 1998) the rate is highly variable (Van der Auwera & Reisslein, 2009). Long video streams are also not statistically stationary in time, which causes a problem when attempting to model video input to a network. This variability is accentuated in the H.264/Advanced Video Codec (AVC) (Schwarz, 2007) and it is reported (Van der Auwera et al., 2008) that the variability is accentuated the more so in the Scalable Video (SVC) extension to H.264, with the result that prior smoothing of VBR streams is contemplated. (The reason for increased variability is attributable to the increased number of motion estimation modes in H.264/AVC and in H.264/SVC, the addition of hierarchical B-frames.)
In temporal smoothing, multiple encoded frames are accumulated so that the compressed bitstream can be packetized and sent at a desired average bitrate. This form of traffic shaping has the disadvantage for video streaming that end-to-end latency is increased by the number of frames accumulated. For ‘conversational’ video services, which have an

additional latency introduced by the need to encode each frame, the effect on the viewer can be disconcerting. Ideally end-to-end latency should be no longer than 200 ms. For this reason, in services such as teleconferencing and videophone, CBR is preferable. However, for pre-encoded video at a significant cost in computational complexity (Salehi et al., 1998) it is also possible through optimal smoothing to send video frames (or rather their compressed bitstream) in advance of their decode time, provided it is known that overflow (or underflow) at the playback buffer will not occur. In the best-effort Internet, jitter introduced by cross-traffic congestion will disrupt these calculations but in those network cores in which ATM or virtual ATM is still in place optimal smoothing has a role. Unfortunately, the presence of access networks of differing types prior to the consumer’s home, or reduced bandwidth links prior to campus and corporate networks introduces an ill-behaved section within the end-to-end path.

Video is known as a delay-sensitive service but in fact there are varying levels of intolerance, and a limit of 200 ms has been mentioned. However, for one-way streaming the delay requirements are less stringent. For example, channel swapping or VCR-like control is restricted to 500 ms intervals, because anchor or key frames at which switching can occur are placed at these intervals within a stream. Another form of delay is start-up delay, with Video-on-Demand (VoD) services hoping to make this imperceptible (< 20 ms), which is perhaps possible on the Internet if the Resource ReSerVation Protocol (RSVP) (Zhang et al., 1997) were to be widely deployed. Variation in delay (jitter) is also important in terms of media synchronization (between audio and video) (Blakowski & Steinmetz, 1996). However, there are also display deadlines to be met, implying that a jitter buffer should be dimensioned to absorb any variation in delivery (assuming Internet delivery). For reference frames (one used for predictive motion estimation), their data is still of value for decoding future frames even if they miss their display deadline. Too large a receiver buffer will lead to increased end-to-end latency and start-up delay, while too small a buffer may cause overflow. This is why adaptive buffers have been contemplated in the research literature (Kalman et al., 2002).

Video streaming is also known as a loss-tolerant service. However, this is misleading as the loss of more than 10% of packets will generally lead to a noticeable deterioration in the quality of the video unless: error-resilience measures have been taken; error control through some form of acknowledgements (ACKs) is used (as in the Windows Media system); Forward Error Correction (FEC) is in place; or error concealment can be applied. A combination of these methods is preferable as part of an error response strategy and unequal error protection (UEP) is possible. In UEP, protection is prioritized according to compressed video content or the structure of the video. Acknowledgments are possible but their impact on delay must always be judged. For example, in (Mao et al., 2003) layered streaming was attempted across an ad hoc network in which multi-hop routing and broken links can lead to high levels of delay. In layered streaming (Mao et al., 2003), a more important base layer allows a basic reconstruction of the video while one or more enhancement layers can improve the quality. However, because of the high risk of delay, in (Mao et al., 2003) it was only possible to send one ACK at most to secure the base layer.

Though FEC schemes with linear decoder complexity (Raptor codes, a variety of rateless erasure codes) have been developed (Shokrollahi, 2006), FEC generally leads to delay in encoding. Because of the additional delay involved in sending acknowledgments (or negative acknowledgments), when there is a long round-trip-time careful engineering needs to be applied if rateless erasure coding is to be used. In rateless or Fountain coding (MacKay, 2005), additional redundant data can always be generated, while in conventional forms of channel coding such as Reed-Solomon, there is a threshold effect whereby if the channel noise or packet erasures pass the level of protection originally provided then all data are lost.

Error resilience techniques, the range of which have been expanded in the H.264 codec (Wenger, 2003), are based on source coding. Error resilience results in lower-delay and as such is suitable for real-time, interactive video streaming, especially video-telephony and video conferencing. However, due to the growing importance of broadband wireless access networks, error resilience is also needed to protect video streaming to the home. This is because physical-layer FEC is already present and, therefore, application-layer FEC may duplicate its role. The exception is if application-layer FEC can be designed to act as an outer code after inner coding at the physical layer, in the manner of concatenated channel coding.

Compressed frame data is often split into a number of slices each consisting of a set of macroblocks. In the MPEG-2 codec, slices could only be constructed from a single row of macroblocks. Slice resynchronization markers ensure that if a slice is lost then the decoder is still able to continue with entropic decoding. Therefore, a slice is a unit of error resilience and it is normally assumed that one slice forms a packet, after packing into a Network Abstraction Layer unit (NALU) in H.264. Each NALU is encapsulated in a Real Time Protocol (RTP) packet. Consequently, for a given frame, the more slices the smaller the packet size and the less risk of packet loss through bit errors.

In H.264/AVC, by varying the way in which the macroblocks are assigned to a slice (or rather group of slices), Flexible Macroblock Ordering (FMO) gives a way of reconstructing a frame even if one or more slices are lost. Within a frame up to eight slice groups are possible. A simple FMO method is to continue a row of macroblocks to a second row, Figure 1a, but allow disjoint slice groups (Lambert et al., 2006). Regions of interest are supported, Figure 1b. Checkerboard slice group selection, Fig, 1c allows one slice group to aid in the reconstruction of the other slice group (if its packet is lost) by temporal (using motion vector averaging) or spatial interpolation. Assignment of macroblocks to a slice group can be general (type 6) but the other six types pre-define an assignment formula, thus reducing the coding overhead from providing a full assignment map.

Data partitioning in H.264/AVC separates the compressed bitstream into: A) configuration data and motion vectors; B) intra-coded transform coefficients; and C) inter-coded coefficients. This data form A, B, and C partitions which are packetized as separate NALUs. The arrangement allows a frame to be reconstructed even if the inter-coded macroblocks in partition C. are lost, provided the motion vectors in partition A survive. Partition A is normally strongly FEC-protected at the application layer or physical layer protection may be provided such as the hierarchical modulation scheme in (Barmada et al., 2005) for broadcast TV. Notice that in codecs prior to H.264, data partitioning was also applied but no separation into NALUs occurred. The advantage of integral partitioning is that additional resynchronization markers are available that reset entropic encoding. This mode of data partitioning is still available in H.264 and is applied to I-frames.

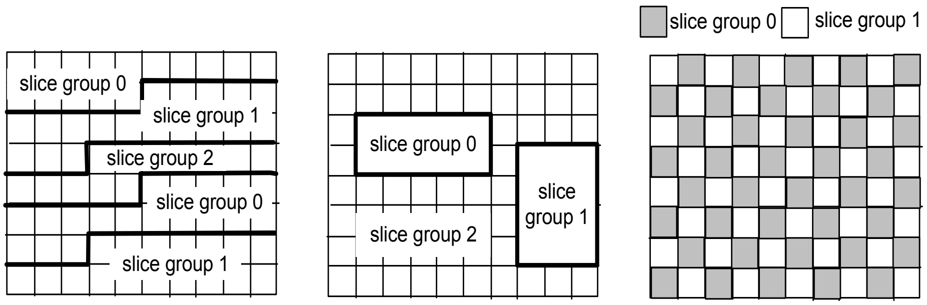


Fig. 1. Example FMO slice groups and types (after (Lambert, 2006) a) Continuing row (type 0) b) geometrical selection (type 2) c) checkerboard selection (type 1)

The insertion of intra-coded macroblocks into frames normally encoded through motion-compensated prediction allows temporal error propagation to be arrested if matching macroblocks in a previous frame are lost. Intra-refresh through periodic insertion of I-frames with all macroblocks encoded through spatial reference (intra-coded) is the usual way of catching error propagation. However, I-frames cause periodic increases in the datarate when encoding at a variable bitrate. They are also unnecessary if channel switching points and VCR functions are not required.

This brief review by no means exhausts the error-resilience facilities in H.264, with redundant frames, switching frames, and flexible reference frames also considered in (Stockhammer & Zia, 2007). We have referred to H.264/AVC anchor frames as I-frames for consistency with previous codecs. In fact, H.264 uses Instantaneous Decoder Refresh (IDR)-frames for the same purpose, whereas H.264 I-frames allow motion estimation reference beyond the Group of Pictures boundary.

Error concealment (Wang & Zou, 1998) is the process of concealing errors at the decoder. However, the form of error concealment is implementation dependent because of the complexity of these algorithms. In fact, for reasons of speed, previous frame replacement is often preferred. If lost frames are replaced by the last frame to arrive successfully there is a danger of freeze frame effects. When there is rapid motion or scene cuts then partial replacement of macroblocks from the previous frame will result in obvious blocky effects. For error concealment in H.264/AVC (Vars & Hannuksela, 2001) the motion vectors of correctly received slices are computed if the average motion activity is sufficient (more than a quarter pixel). Research in (Vars & Hannuksela, 2001) gives details of which motion vector to select to give the smoothest block transition. It is also possible to select the intra-coded frame method of spatial interpolation, which provides smooth and consistent edges at an increased computational cost. Experience shows a motion-vector-based method performs best except when there is high motion activity or frequent scene changes (Kim & Kim, 2002).

2.2 Streaming systems

In networked video delivery, systems are classically divided (Chou, 2007) into streaming and broadcast systems. In the former, video is pre-encoded before storage and access by a server, while in the latter there is no storage before server access and multicast over a network. A further distinction in this model is that in streaming a control path exists, whereas the presence of many receivers in a broadcast system means that feedback would be impossible to manage. Feedback can be used for congestion control but it can also return VCR commands, typically through the Real Time Streaming Protocol (RTSP) (Schulzrinne et al., 1998). Nevertheless, it is possible to stream both pre-encoded and online or live video because, after feedback notification of congestion, the streaming rate can be changed through bitrate transcoding (Assunção & Ghanbari, 1997) (Sun et al., 2005). One problem that fast transcoding may face in the H.264 codec is error drift when transcoding I-frames (Lefol et al., 2006).

Scalable video also allows rate control as a response to network conditions or target device capability but a full discussion of the variety of multi-layer or scalable options such as Fine Grain Scalability (Radha et al., 2001), Multiple Description Coding (Wang, 2005), signal-to-noise ratio (SNR) scalability (Pesquet-Popescu et al., 2006) would require another chapter. Rich though the scalable options are commercial Internet operators seem to prefer simple schemes such as simulcast as used by RealVideo. In simulcast, multiple streams are stored (or encoded online) at different rates and selected according to network conditions. In H264/AVC, stream switching frames allow a smoother transition between low and higher quality stream at lower cost in bandwidth than through switching at I-frames.

At the target device, video is first buffered in a playout buffer, decoder or client buffer (there are various alternative names) prior to access by the decoder. This buffer will vary in size depending on the capabilities of the device. Large buffers are not advisable for battery-powered devices because of both active and passive energy consumption. Nevertheless some buffering is required to absorb variation of delay (jitter) over the network.

Because of motion-compensated prediction coding it is always necessary to store packets prior to decode, especially if VBR is in use. An additional render buffer, able to store a few frames prior to display, is also generally present. Apart from buffer overflow in the intermediate buffers of routers through congestion, buffer overflow at the playout buffer is also possible. Packets arriving too late for their display or decode deadlines may also be dropped. It is also possible, because of jitter, for buffer underflow to occur. In fact, in the Windows Media system (Chou, 2007) the receiver monitors the buffer level to detect network congestion. Again like RealVideo, Windows Media uses simulcast, with the receiver signaling the server to swap to a lower rate stream when it detects congestion. However, the Windows Media receiver or client is not only reliant on buffer monitoring, because packet loss at the receiver is also taken into account.

3. Congestion control

In this Section, the focus is on congestion control of single stream unicast for IPTV and other multimedia services. Because the main thrust in congestion control research is to provide an enhanced service through VBR delivery, this Section concentrates on that whereas in Section 4 on statistical multiplexing, multi-channel delivery of CBR streams is considered. The latter is likely to be a broadcast service.

3.1 IPTV and unicast streaming

Real-time video applications, such as IPTV, video-on-demand (VoD), and network-based video recorder interest telecommunication companies, because of their high bitrates, though they also risk overwhelming existing networks if it is not possible to control their flows. The unicast variety of IPTV is very attractive because it allows streaming of individual TV programs at a time chosen by the end user. Broadly speaking, two types of heterogeneous delivery network exist: 1) the familiar Internet, with best-effort Internet Protocol (IP) routing, i.e. an unmanaged IP network; and 2) All-IP networks, which retain IP packet framing but, particularly in the network core, switch packets (across Clos switches) rather than employ packet routers, i.e. a managed IP network. These IP networks are generally referred to as converged networks, as they combine a traditional telephone service (through Voice-over-IP) with data delivery (normally high speed Internet access) and TV (through IPTV). The marketing term for such a combined service is 'triple-play' and if mobility is added then this term becomes 'quadruple-play'.

IPTV services are in active commercial development for converged telephony networks, such as British Telecom's 21st Century Network (21CN) (Geer, 2004) or the all-IP network of KPN in the Netherlands.. Within the 21CN, video streaming is sourced either from proprietary servers or from an external Internet connection, with best-effort routing. Before distribution from the server to individual users, multiple videos streams will share a multimedia channel, an example being MPEG-2 Transport Stream which serves for H.264/AVC pre-encoded streams. These video streams could represent different TV channels that can be selected by the IPTV user. However, when the multimedia channel leaves the core network it is commonly delivered across an access network such as Asymmetric Digital Subscriber Line (ADSL) (Zheng & Liu, 2000), when different delivery conditions apply.

On the Internet, video streams must coexist with other data traffic, while in emerging All-IP networks multimedia traffic may predominate. In an All-IP network, as in the Internet, a capacity restriction may still exist at the connection between the network core and the access network, of which the technology can be cable (Vasudevan et al., 2008), broadband wireless (IEEE, 2004), or connections to the Video Serving Office (Han et al., 2008) from which video is typically distributed over Asymmetric Digital Subscriber Line (ADSL) connections. Note also that Internet traffic may be directed through an All-IP network by means of the common agency of IP framing.

In the Internet, a tight link (or more loosely a bottleneck), which commonly exists at the network edge before a corporate or campus network (Cisco, 2000), is the link of minimum available bandwidth on a network path. Strictly the term 'bottleneck' defines the bandwidth capacity of a network path, which while the path exists is a constant, though the term may also be loosely applied to a tight link. A tight link is a dynamic concept, as its location will vary firstly over time according to background traffic patterns and secondly according to the network path's route, which is not fixed because of dynamic routing on the Internet. These two factors can create uncertainty in any video streaming response. Available bandwidth is restricted by coexisting cross-traffic, which is most likely carried by the Transmission Control Protocol (TCP) and predominantly originates from web-servers or P2P file transfer (Xie et al., 2007). Transport-layer protocols like TCP, sitting above IP, are responsible for end-to-end negotiation of delivery between applications. On All-IP networks, coexisting traffic across a network sub-channel or pipe is more likely to arise from other proprietary video servers and be carried by the minimal User Datagram Protocol (UDP) as directed by congestion controllers. A pipe is a virtual bandwidth restriction imposed by quality-of-service requirements that must balance the requirements of other types of traffic and the capacity of the access network. As in the Internet, All-IP congestion controllers should be end-to-end over the network path, allowing a general solution in the sense that the nature of the access network bottleneck may not be known in advance. In an All-IP network, statistical multiplexing of VBR video sources within a video pipe may increase its efficiency but there is no spare capacity for greedy acquisition of bandwidth by independently controlled video servers. We return to the subject of statistical multiplexing within the IPTV pipe in Section 4.

Congestion control is vital to avoid undue packet loss from the fragile compressed video stream. At the sub-frame level, because variable-length coding (VLC) prior to outputting the bitstream introduces a dependency between each encoded symbol, there is fragility that error resilience techniques such as decoder synchronization markers and reversible VLC only partially address. Because successive video frames are broadly similar (except at scene cuts and changes of camera shots), only the difference between successive frames is encoded in order to increase coding efficiency. Consequently, at the frame-level, removing temporal redundancy introduces a dependency on previously transmitted data that implies lost packets from reference frames will have an impact on future frames.

Unicast video streaming, which brings increased flexibility and choice to the viewer over multicast delivery, is achieved by determining the available bandwidth and adapting the video rate at a live video encoder or an intermediate transcoder. Fuzzy Logic Control (FLC) is suited to congestion control (Jammeh et al., 2007), because of the inherent looseness in the definition of congestion and the uncertainty in the network measurements available, together with the need for a real-time solution. Within video coding it has previously found an application (Grant et al., 1997) in maintaining a constant video rate by varying the encoder quantization parameter according to the output buffer state. This is a complex control problem without an analytical solution. Fuzzy logic is gaining acceptance in the video community, witness (Rezaei et al., 2008), but it turns out that further improvements are possible with interval type-2 (IT2) fuzzy logic.

3.2 Fuzzy logic control for congestion

In our application, FLC of congestion is a sender-based system for unicast flows. The receiver returns a feedback message indicating changes to the delay experienced by video stream packets crossing the Internet. This allows the sender to compute the network congestion level and from that the FLC estimates the response. The same controller also should be able to cope with a range of path delays and with video streams with differing characteristics in terms of scene complexity, motion, and scene cuts.

Traditional, type-1 FLC is not completely fuzzy, as the boundaries of its membership functions are fixed. This implies that there may be unforeseen traffic scenarios for which the existing membership functions do not suffice to model the uncertainties in the video stream congestion control task. IT2 FLC can address this problem by extending a Footprint-of-Uncertainty (FOU) on either side of an existing type-1 membership function. In IT2 fuzzy logic, the variation is assumed to be constant across the FOU, hence the designation `interval'. Though the possibility of type-2 fuzzy systems has been known for some time (Zaddeh, 1975), only recently (Mendel, 2007) have algorithms become available to calculate an IT2 output control value at video rate. The first IT2 controllers (Hagras, 2007) are now emerging, in which conversion or retyping from fuzzy IT2 to fuzzy type-1 takes place before output. For video streaming there are important practical advantages. Not only does such a controller bring confidence that re-tuning will not be needed when arriving traffic displays unanticipated or un-modeled behavior but the off-line training period required to form the membership functions can be reduced.

We now compare type-1 FLC for congestion control of video streaming to an IT2 FLC and compare the performance in the presence of measurement noise that is artificially injected to test the relative robustness. The delivered video quality in terms of Peak Signal-to-Noise Ratio (PSNR) is equivalent to the successful type-1 FLC when the measurement noise is limited and under test results in a considerable improvement when the perturbations are large. We go on to compare the IT2 FLC to a non-adaptive approach and to congestion control by two well-known controllers, TCP-friendly Rate Control (TFRC) (Handley et al., 2003) and TCP Emulation at Receivers (TEAR) (Rhee et al., 2000), one sender-based and the other receiver based. These are tested by their ability to support multiple broadband connections over an all-IP network. However, firstly we introduce fuzzy logic control.

3.3 Fuzzy logic control

Figure 2 is a block diagram of FLC of congestion, with two inputs, the packet delay factor, *df*, and delay samples to form a trend (whether packet delay is increasing or decreasing). The formation of these inputs is described in Section 3.4. These inputs are converted to fuzzy form, whereby their membership of a fuzzy subset is determined by predetermined membership functions. This conversion takes place in the fuzzifier and trend test units of Figure 1. The fuzzy outputs are then combined in the inference engine through fuzzy logic. Fuzzy logic 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. The control signal causes the quantization parameter of the video stream to be changed, thus adjusting the output bitstream.

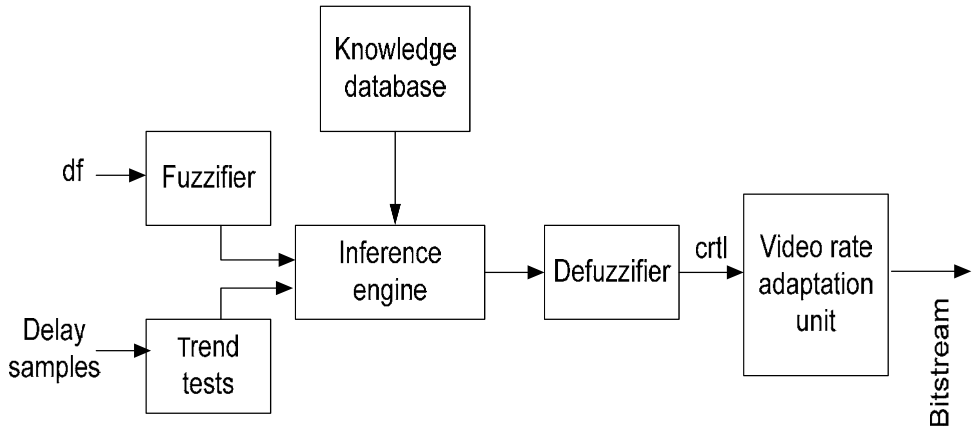


Fig. 2. FLC delay-based congestion controller

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.
The FLC determines incipient congestion from one way packet queuing delay in intermediate router buffers. 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. For each received packet indexed by *i*

$$OWDi = Tr - Ts, \tag{1}$$

where *Tr* is the receive time of the current packet and *Ts* is the time the packet was sent. When it is appropriate, the computed *OWDi* updates the minimum and maximum one-way delays (*OWD*s), *OWDmin* and *OWDmax*, on a packet-by-packet basis. Subsequently, the maximum queuing delay is found as $maxQD = OWDmax - OWDmin$.

The queuing delay over the network path, *QDi* is computed from the measured delay and the minimum delay:

$$QDi = OWDi - OWDmin \tag{2}$$

and an exponentially-weighted average of the queuing delay for the ith received packet is formed by,

$$avgQDi = (1 - a) \times avgQDi{-}1 + a \times QDi \tag{3}$$

where $\alpha \leq 1$ is a forgetting constant. In tests, *a* was set to 0.1. A delay factor, Df , is computed from the average queuing delay and the maximum queuing delay,

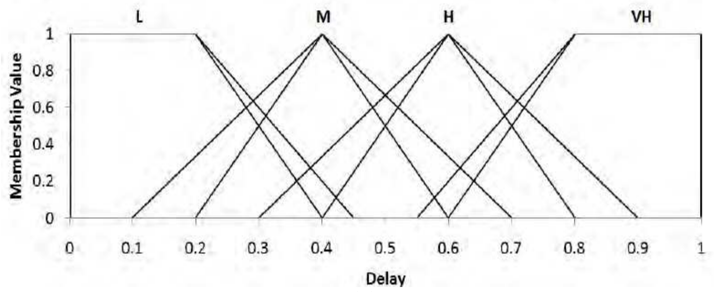
$$Df = avgQDi\,/maxQD \tag{4}$$

where *Df* ranges between [0,1] with 0 indicating no incipient congestion, 1 indicating full-blown congestion, with shades of incipient congestion between 0 and 1. *Df* is an early notification of congestion and is the first input to the FLC.
A trend analysis method is used to determine the general trend of the average delay. In each measurement epoch, a number *k* of queue delay samples are grouped into *τ* groups where $\tau = \sqrt{k}$. We use the pairwise comparison test (PCT) to determine the overall trend of the queueing delay as shown in (5).

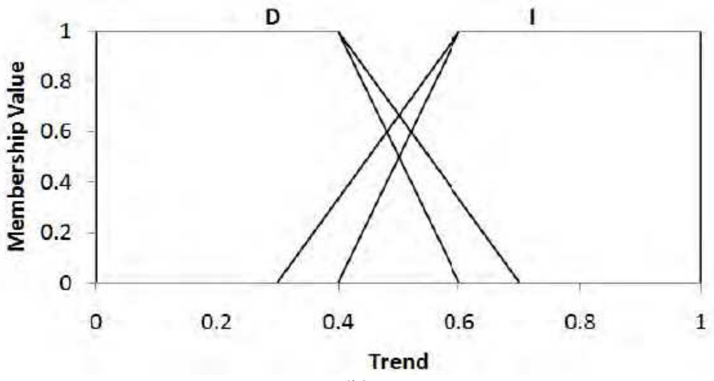
$$T_{PCT} = \frac{\sum_{i=2}^{\tau} I(M^{i} > M^{i-1})}{\tau - 1} \tag{5}$$

where M^i is the median of group *i* and *I(X)* is 1 if *X* holds and 0 otherwise. The value of T_{PCT} is sent back to the sender where a fuzzifier determines whether the level was increasing or not according to a membership function.

IT2 input membership functions for *Df* and trend are constructed, Figure 3, as an extension of the type-1 FLC through an FOU at the boundaries of the formerly crisp (fixed) membership functions. Assuming the usual singleton input of *Df* (or T_{PCT}), an interval set requires just an upper and lower value to be resolved to form the resulting FOU in the corresponding output set. For example, Figure 4 shows two IT2 membership functions for input sets A and B, each with an FOU. Singleton input X is a member of each with different degrees of membership. Strictly, an infinite number of membership functions (not all necessarily triangular) can exist within the FOUs of sets A and B, but IT2 sets allow the upper and outer firing levels to be taken, as shown in Figure 4. The minimum operator (min) acts as a *t*-norm on the upper and lower firing levels to produce a firing interval.
The firing interval serves to bind the FOU in the output triangular membership function shown to the right in Figure 4. The lower trapezium outlines the FOU, which itself consists of an inner trapezoidal region that is fixed in extent. The minimum operator, also used by us as a t-norm, has the advantage that its implementation cost is less than a product t-norm. (A *t*-norm or triangular norm is a generalization of the intersection operation in classical logic.) Once the FOU firing interval is established, Center-of-Sets type reduction was applied by means of the Karnik-Mendel algorithm, which is summarized in (Mendel, 2007). Type reduction involves mapping the IT2 output set to a type-1 set. In practice, defuzzification of this type-1 output fuzzy set simply consists of averaging maximum and minimum values. The result of defuzzification is a crisp value that determines the change in the video rate.



(a)



(b)

Fig. 3. IT2 FLCC (a) Delay factor (Df) (b) Trend membership functions.

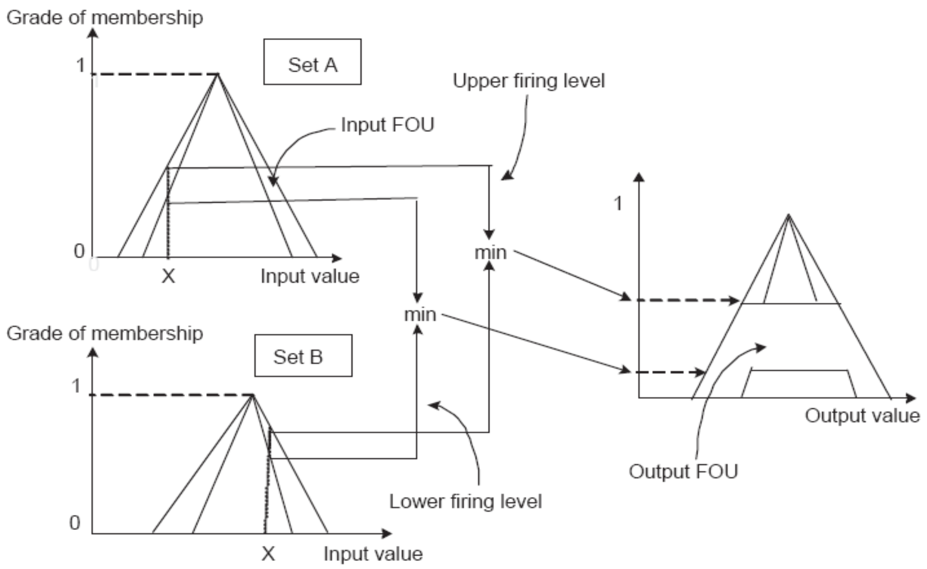


Fig. 4. IT2 FL calculation of output FOU

Figure 5 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 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 bitrate transcoder adapting pre-encoded video or an encoder adapted through its quantization parameter) changes the sending rate to that computed by the fuzzy controller. The current implementation changes the quantization level of a frequency-domain transcoder (Assunção & Ghanbari, 2000) for VBR video. Full decode and re-encode is prohibitively time consuming. Prior approaches relied on estimation of the error introduced by re-quantization without taking account of the impact on motion estimation by reconstructing the picture and reusing information in the bitstream (Vetro et al., 2005), which still introduces delay, whereas partially (entropic) decoding and motion estimation in the transform domain is faster.

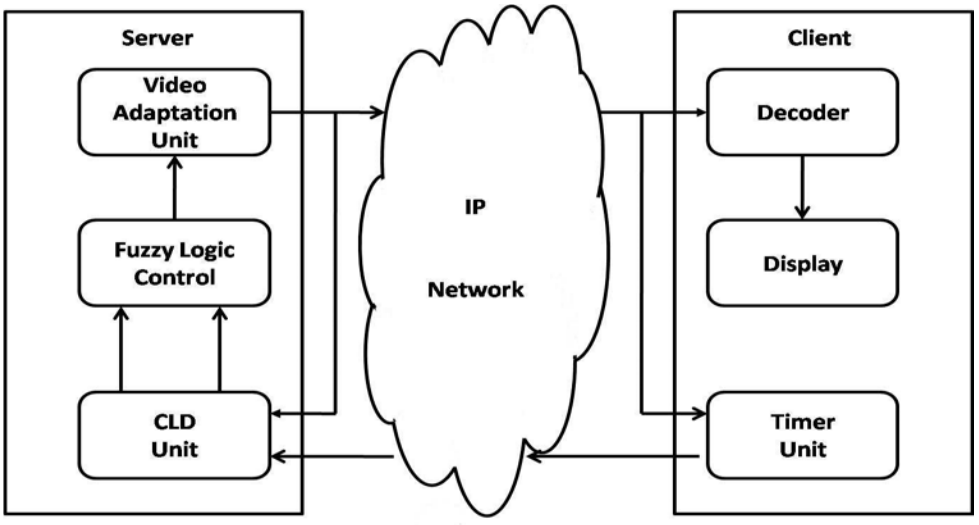


Fig. 5. Video server for all-IP network

Figure 6 shows one instance of server and client. VoD mode, IPTV or video clip services there are multiple video streams and multiple clients. Figure 6 assumes a bank of such servers delivered over an access network such as ADSL or ADSL2+, with downstream rates to 24 Mbps and beyond, one of the passive optical network types (PON) terminating in 100 Mbps Ethernet or coaxial cable, or broadband wireless such as IEEE 802.16 (WiMAX).

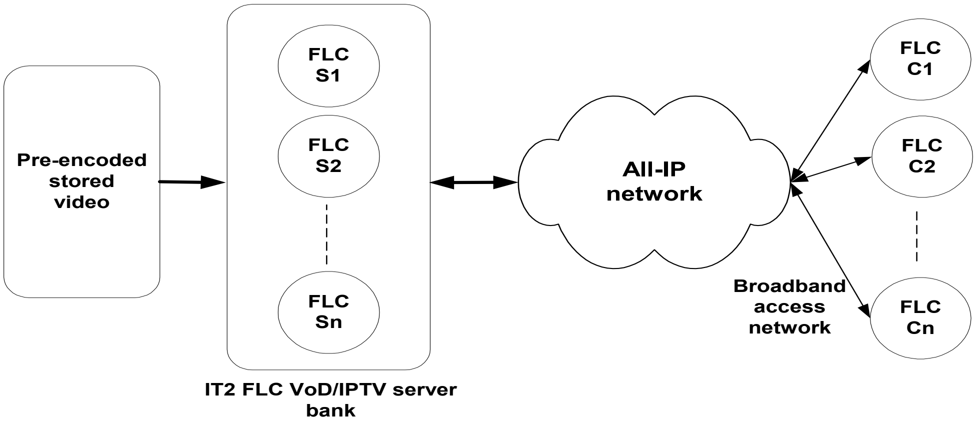


Fig. 6. VoD IPTV video delivery architecture

3.4 Evaluation

FLC congestion controller employs delay and its variation to gauge the state of the network. There is, however, inherent noise in the measurement of delay, including packet timestamps with limited resolution and unresolved clock drift between sender and receiver. These uncertainties in the input to an FLC will potentially impact its performance.

The well-known ns-2 network simulator (v. 2.32) was used, with the type-1 and IT2 FLC implemented as new protocols within ns-2. A normal distribution generated a random noise value with zero mean and a specified standard deviation, determined by the level of noise required and dynamically adjusted relative to the measured (simulated) value. For each simulation the level of additional noise was incrementally increased. At each incremental step, the performance of the two controllers was compared in terms of rate adaptation accuracy, packet loss rate, and delivered video quality (PSNR). Input was a 40 s MPEG-2 encoded video clip, showing a newsreader with a changing backdrop, with moderate movement. The VBR 25 frame/s Standard Interchange Format (SIF)-size clip had a Group of Pictures (GOP) structure of N=12, M=3 where N is the number of pictures (frames) between each Intra-coded picture and M is the number of pictures between each prediction-coded picture within the GOP (Ghanbari, 2003). For error resilience purposes, there was one slice per packet, resulting in 18 packets per frame. The FLC controllers adjusted their rate every frame. In this set of tests the encoded video was stored at a mean rate of 1 Mbps. The video streams were passed across a bottleneck link restricted to 400 kbps in capacity.

The results are gathered in Table 1, and Figs. 7–8. Below 30% additional noise, the two controllers do not significantly deviate. However, beyond 30% of additional noise, the IT2 FLC congestion controller showed significant improvement over the type-1 FLC in terms of reduced fluctuation in the sending rate and a reduced packet loss rate, both of which will be reflected in better average delivered video quality. The smoothness of the transmission rate (measured by a reduction in the standard deviation of the delay on a per-packet basis) is important in video transport as a fluctuating compressed bit-rate implies a fluctuation in video quality, which is more disconcerting to a viewer than a stream of consistent quality, even if that average quality was lower than that of a fluctuating stream. Figure 8 confirms that delivered average video quality is improved, though, for very high levels of measurement noise, the encoded video stream is so corrupt it matters little which FLC is in control, the quality is very poor. Detailed statistical examination of these results has confirmed their significance within 90% confidence intervals.

Noise level (%)	**Type-1**	**Type-2**
0	77.527	76.722
10	78.192	76.607
20	78.986	77.098
30	80.281	77.677
40	109.927	77.747
50	193.612	78.244
60	227.173	80.238
70	230.016	84.294
80	230.651	93.822
90	230.924	113.355
100	231.082	124.652

Table 1. Standard deviation of FLC type-1 and type-2 sending rates (kbps)

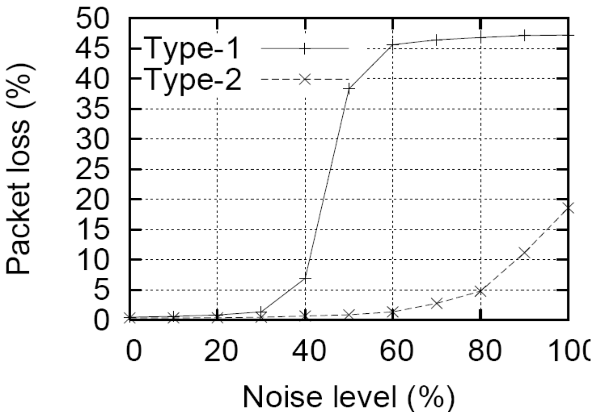


Fig. 7. Packet loss rate for an increasing noise level

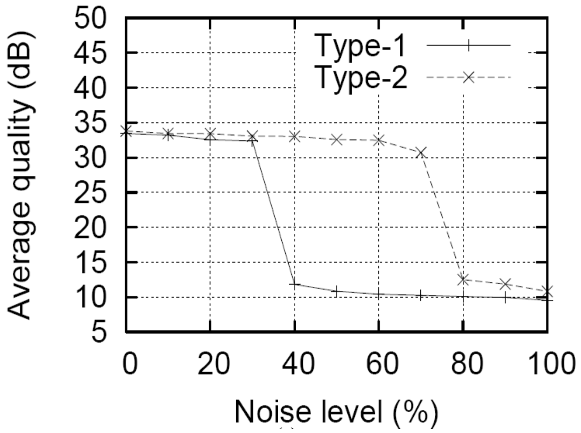


Fig. 8. Mean received video quality for an increasing noise level

Comparison was also made with the TFRC protocol, the subject of an RFC (Handley et al., 2003) and a prominent method of congestion control. The intention is that the average rate of TFRC should be equivalent to the dominant protocol in the Internet, TCP. However, the short term TFRC rate is intended to be less aggressive than TCP as sharp fluctuations in coding rate will result in variable quality at the receiver. In that way, it is hoped that TFRC will avoid causing congestion collapse by greedy acquisition of bandwidth. In TFRC, the sending rate is made a function of the measured packet loss rate during a single round-trip time (RTT) duration measured at the receiver. Unfortunately, if the TFRC feedback frequency is reduced TFRC tends to dominate co-existing flows (Rhee et al., 2000). The sender then calculates the sending rate according to the TCP throughput equation given in (Handley et al., 2003). As with IT2 FLC and TEAR, the UDP transport protocol is employed to avoid unbounded delays, which are possible with TCP transport.

No control			
No. of Sources	**Loss rate (%)**	**Link use (%)**	**PSNR (dB)**
25	0.0	100.0	–
30	16.66	120.0	–
35	28.56	140.0	–
40	37.49	160.0	–
45	44.44	180.0	–
50	49.99	200.0	–
TFRC			
No. of Sources	**Loss rate (%)**	**Link use (%)**	**PSNR (dB)**
25	1.50	101.48	36.08
30	1.81	101.80	35.11
35	2.11	102.80	33.78
40	2.39	102.44	33.07
45	2.65	102.78	31.34
50	2.91	102.96	30.18
TEAR			
No. of Sources	**Loss rate (%)**	**Link use (%)**	**PSNR (dB)**
25	2.50	102.52	33.27
30	3.51	103.60	32.34
35	4.61	104.80	31.56
40	5.75	106.08	30.70
45	6.86	107.36	29.61
50	7.91	108.56	28.78
IT2 FLC			
No. of Sources	**Loss rate (%)**	**Link use (%)**	**PSNR (dB)**
25	0.0	89.82	39.61
30	0.0016	99.96	37.90
35	0.0026	99.96	36.89
40	0.0029	99.96	35.44
45	0.0038	99.84	33.19
50	0.0048	99.82	31.40

Table 2. Performance comparison of congestion controllers

Unlike TFRC, TEAR is based on the Arithmetic Increase Multiplicative Decrease (AIMD) algorithm of TCP. Unlike TCP, TEAR avoids the oscillatory behavior of TCP by averaging its sending rate over a round, based on the time to send a congestion window's packets. TEAR's sending rate approximates that of an equivalent TCP source. Both TFRC and TEAR rely on measurements of the RTT, while TFRC is also adversely affected by inaccurate loss rate estimates (Rhee et al., 2007). Without a transcoder TFRC and TEAR require playout buffers to smooth out network delay. Therefore, PSNR is affected by loss rate only, assuming a large enough buffer to avoid overflow. FLC also reduces the video quality through transcoding if there is insufficient bandwidth, but this avoids the need for long start-up delays and allows smaller buffers on mobile devices. In further comparison tests, the standard 'dumbbell' network topology was assumed with a bottleneck of 25 Mbps. The one-way delay, modeling the latency across the complete network path, was set to 40 ms, which is the same as the maximum delay across a country such as the U.K or France. Side link delay was set to 1 ms and the side link capacity was set to easily cope with the input video rate. The mean encoded video rate was again 1 Mbps. The buffer size on the intermediate routers was set to RTT$\times$bandwidth, to avoid overflow through too small a buffer. The router queuing discipline was drop-tail. The intention of these tests was to see how many video streams could be accommodated across the bottleneck link. In Table 2, the number of controlled video sources was incrementally increased.

The starting times of streaming the 'news clip' to each client was staggered, and then each clip was repeatedly sent over 200 ms. The first 40 s of results, was discarded as representing transient results. This method was chosen, rather than select from different video clips, because the side effects of the video clip type do not intrude.

As can be seen from Table 2, when there is no control, there is no packet loss until the capacity of the link is reached. Thereafter, the link utilization grows and, as might be expected, the packet loss rate rapidly climbs. Failure to estimate the available bandwidth causes both TFRC's and TEAR's mean link use to exceed the capacity of the bottleneck link. As the number of flows increases, it becomes increasingly difficult to control the flows and there is a steady upward trend in the overshoot. In respect to TEAR, this leads to considerable packet loss. The packet loss patterns are reflected in the resulting PSNRs, though there is no direct relationship because of the effect of motion estimation in the codec. It is surprising in that TEAR was developed after TFRC and in part as a reaction to it (Rhee et al., 2007). However, subsequent to the development of TEAR, TFRC has undergone some refinements such as TCP's self-clocking. However, from Table 2 it is apparent IT2 FLC congestion control does not suffer from the difficulties that TFRC and TEAR encounter. There is a very small loss rate due to moments when the time varying nature of VBR video results in the FLC overestimating the available bandwidth but this is significantly below the loss rates of the traditional controllers.

4. Statistical multiplexing

4.1 IPTV and statistical multiplexing

Fortunately, compressed video streams forming the TV channels making up the IPTV service will not necessarily have the same bandwidth requirements, as their content complexity will vary over time with changes in their spatial and temporal complexity. In the long term, for entertainment applications this variation is determined by the video genre, such as sport, cartoon, 'soap' and so on but there are also changes over a shorter time period caused by such factors as the type of video frame and whether there is a shot change or a scene cut. Consequently, multiple video streams as part of an IPTV service can each be adaptively allocated a proportion of the bandwidth capacity according to their content complexity.

As IPTV bandwidth may be constrained by a particular access network technology a practical solution, which has already been developed in the UK and Japan (Kasai et al., 2002), is to employ a transcoder bank to change the rates of video streams within the