Congestion-Aware Scalable Video Streaming

Nuruddeen Iya, Fabio Verdicchio, Gorry Fairhurst
Electronics Research Group
University of Aberdeen
Aberdeen, UK
(nuruddeeniya, fverdicc, g.fairhurst)@abdn.ac.uk

***Abstract*— Real-time media applications often ignore ongoing congestion if there is no option to reduce the rate. These applications pose a threat to themselves and other traffic. Reducing the transmission rate requires reducing the amount of packets rather than spreading the transmission over a longer interval. Loss-based congestion control mechanisms are unsuitable for this requirement. Also this rate reduction with popular video codecs, e.g. MPEG4, is often problematic. This paper investigates the problems associated with real-time video transmission over the Internet. We investigate a rate control method minimizing delay and losses and report preliminary but promising results.**

I. Context and Motivation

The Internet has in recent years seen an unprecedented growth of live video streaming applications. Multimedia applications that compete with other traffic for limited network resources, can inject more data than a network path can sustain. Eventually buffers fill, resulting in delay and loss. Traffic flows that share the same router queue are disrupted: existing traffic is starved of resources and video quality is impaired. Multimedia applications using a potentially congested path require congestion control to effectively share network capacity.

Interactive network multimedia has expectations in terms of loss and a desire to avoid excessive data. For many applications the rate at which traffic is injected into the network is only limited by the video codec. To mitigate the effects of loss, packet forward error correction (FEC) may be added. It would be beneficial to all other network users to adapt to the available capacity and avoid delay or losses.

There is renewed interest in the use of web-based interactive conferencing under the RTCweb initiative. The IETF RMCAT [1] Working Group is defining congestion control methods appropriate to multimedia.

II. Congestion Control for Real-Time Video

Real-time media imposes a strict constraint on latency experienced by traffic: (*i*) late packets at the receiver are discarded; (*ii*) there is no time for retransmission of lost packets; (*iii*) a missing packet carrying part of a reference frame (on which other frames depend) impairs video reconstruction.

TCP congestion control [2] was introduced to the Internet by Van Jacobsen in 1986 to avoid congestion collapse. The original method has evolved into a sophisticated set of techniques that make TCP flows "reactive" to congestion signals, i.e. packets that are marked or dropped from the network path. While this has stabilized the Internet for over 20 years, there are issues with this approach and the methods are not sufficient to provide good service in all circumstances.

Early attempts to use reactive loss-based approaches like TCP incurred significant delay due to the following: TCPs design that optimizes throughput and induces queuing delay at the bottleneck; TCP's loss-recovery mechanism that requires in-order delivery to the application; and TCPs socket interface that uses a stream-based delivery model. Some issues may be fixed by redesigning TCP to better support multimedia [22].

Another approach was adopted in TCP-Friendly Rate Control, TFRC [3]. This uses an equation-based model to set an upper limit to the capacity a flow can transmit, and enables applications to avoid the delay of retransmission.

While transport methods can add delay, they are not the only contributor impacting the responsiveness of applications supporting live video streaming, the media codec handling is also important: packets encoding a one-second video segment must be received over a one-second interval; if the delivery time consistently exceeds the segment duration, then the receiver's video lags progressively behind the sender's. Hence reducing the transmission rate requires reducing the volume of packets encoding the video (sacrificing visual quality) rather than spreading transmission of packets over a longer interval.

It is often difficult to change the media rate of popular video codecs such as MPEG2 [4] or MPEG4/AVC [5]. One approach is to halt the encoding process, switch encoder configuration and re-start; even using two encoders in parallel a client can not switch between configurations at any arbitrary point. As an alternative, a sender can selectively discard a (small) fraction of encoded video frames; this avoids the stall, but the resulting rate reduction is modest. These are a key limitation for an efficient reactive congestion control.

It is also important to consider the receiver. Packet loss typically results in a sudden and noticeable reduction in visual quality; while decoder mechanisms can mitigate losses, these need to be avoided to ensure good experience. Delay variation in the network demands that a receiver implements a playout buffer to enable (moderately) delayed packets to be useful.

A bulk TCP congestion control method is unsuited for delay-sensitive applications – this not only induces regular packet loss (as TCP probes for capacity), it also builds a queue at the bottleneck router. This makes loss-based mechanisms undesirable for video congestion control.

978-1-4799-1270-4/13/$31.00 ©2013 IEEE

Delay-based methods, use delay as an implicit metric to indicate the status of the bottleneck queues, reacting to indications of the onset of congestion (e.g. a delay-limit is exceeded) by gracefully reducing the rate before it experiences packet loss. This closely relates to the design constraints for real-time applications: reduce path delay and avoid losses.

Examples of delay-based congestion control include TCP Vegas [6], [7], and LEDBAT [8], CARD [11], CTCP [12], Cx-TCP [13], and [7], or recent proposals such as Sprout [9]. These schemes use proactive signaling, attempting to react before congestion occurs, but have not been designed for multimedia. There is a need to evaluate congestion methods against the requirements of real-time video traffic. The RMCAT WG is therefore evaluating proposals such as DFlow and NADA [1].

As application responsiveness becomes more important, there has been renewed interest in Active Queue Management (AQM). This can prevent excessive queues building in the network, and can enable an alternate proactive approach using ECN [10]. While such approaches are expected to be helpful, the focus of this paper on current technologies cannot assume the existence of AQM or ECN in the network.

III. Scalable Network Video Streaming

We propose a framework for rate-adaptive congestion-controlled network video [14] based on a Scalable Video Codec (SVC) [15]. The framework comprises two components: an SVC-based Rate-adaptation Engine (SRE), and a Congestion and Delay Monitor (CDM) as shown in Fig.1. The key contribution is that our framework combines the adaptation of the video stream to the network (SRE) with a dynamic analysis of the path condition (CDM) to drive the adaptation.

Congestion-controlled video transport requires that the sender adapts the media quality to the available network capacity. Existing research on bitstream adaptation, e.g. [1] [16] [17] [18] drive the rate control by encoding the video source at the best visual quality for a given target rate. The SRE instead uses SVC to encode a video source into a scalable stream targeting a maximum bitrate (hence visual quality). The resulting packets are then grouped into subsets, each subset comprises the same number of packets (hence has the same rate), with each assigned a Priority ID (PID). For instance, creating 10 PIDs corresponds to creating 10 substreams each including 10% of the total data, hence total rate. Each PID, labeled from 1 to 10, progressively contribute to the video quality. When enough capacity is available, all PIDs are transmitted, resulting in full quality. When a rate reduction is required, PIDs are discarded progressively in descending order. This ensures a progressive fine-grained reduction of the rate and a corresponding graceful decrease of the visual quality. Importantly, the rate is adapted *on-the-fly* by transmitting only the appropriate subset of the unique SVC stream, without incurring any additional encoding complexity.

At the sender, the SRE increases or reduces transmission rate driven by the CDM. The first aim is to ensure that the packets encoding the current part of the video, say a one-second segment, are timely received, i.e. all packets arrive by the time the video segment is to be displayed. This delay-based

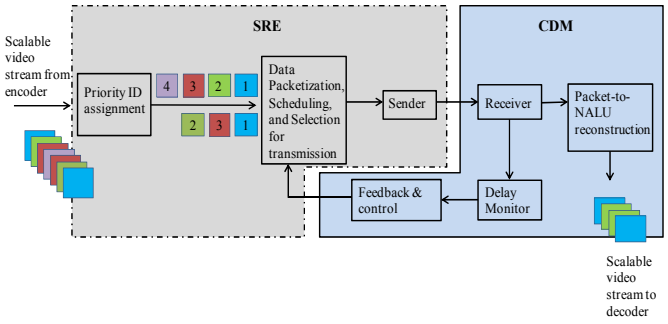

Fig. 1. Scalable Network Video Streaming Framework

method is discussed in the next section. This approach proves effective when the link is shared with delay-regulated flows. Coexistence with TCP flows causes our system to switch to a different rate control technique, as described in section VI.

IV. Delay-based Rate Control

In the absence of packet loss, the main task of the CDM is to avoid building an excessive network queue. The CDM responds to increased delay by having the SRE gracefully reduce the transmission rate (dropping PIDs). In case of losses, a sharp rate reduction is requested to the SRE; the delay associated with a loss event is recorded and then interpreted as a sign of impending congestion. A flow reaching the path capacity for the first time "takes note" and avoids contributing a similar load in the future.

In summary, our delay-based approach uses one-way delay to infer queuing delay. The delay currently experienced by packets is compared to an interval bounded by two extremes: minimum delay $d_{\min}$, and maximum delay $d_{\max}$. The former is the minimum delay experienced by a packet, updated over several reporting intervals as `new_dmin = min(old_dmin, current_delay)`. The latter is set to be the smaller of the following: (*i*) a value proportional to the size of the playout buffer at the receiver; (*ii*) the delay associated with a loss event. The CDM adjusts the rate depending on where the reported delay lies amongst a set of thresholds within the interval $[d_{\min}, d_{\max}]$. Further details given in [14] are omitted here for brevity. Instead we exemplify the system behaviour.

We simulated the network topology shown in Fig.2 with NS-2 [21]. Two video flows, each with nominal rate of 840 kbps, share a bottleneck with 1.2 Mbps capacity. As shown in Fig.3, the two flows begin transmitting at different times; each flow increases its rate linearly, with video1 reaching 8 PIDs at around 120s, while video2 reaches 6 PIDs in the same interval. The subsequent rate increase (by a single PID) for video2 results in a combined flow rate that progressively fills the bottleneck buffer as indicated by the ramp in the delay. This leads to a reduction of rate by both flows at the same time (video1 reduces from 8 to 6 PIDs and video2 from 7 to 5). Then, each flow takes a fair share of the link without hurting the other. No loss was recorded. Visual quality, measured using PSNR [14], is stable around 30 dB for both flows.

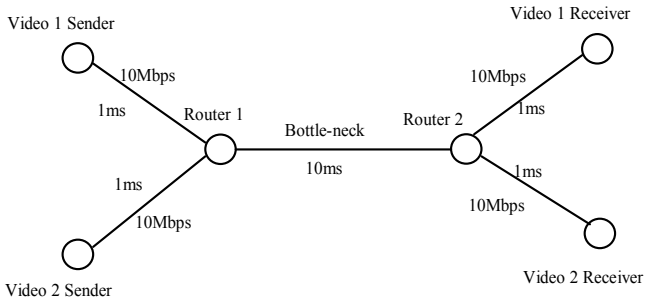

Fig. 2. Simulation Topology

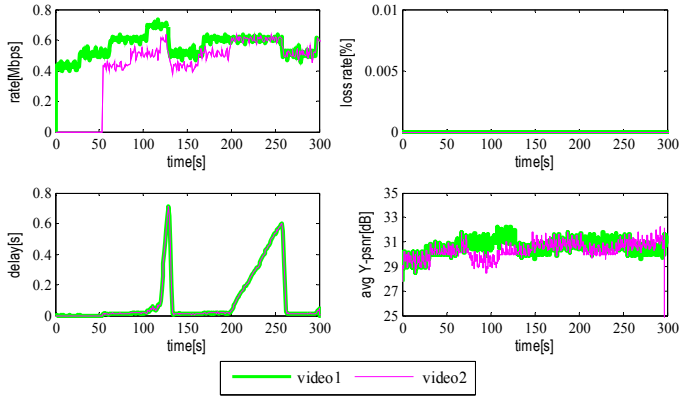

Fig. 3. Video flows sharing a 1.2 Mbps link.

V. Overcoming Limitations of Delay-Based Techniques

This section discusses threats to algorithms relying only on one-way delay, as [14]. We also note that extensive evaluations are now required. Delay and congestion measurement are trivial in simple simulation topologies, it becomes much more difficult in topologies reflecting actual networks, with links with varying capacity (e.g. due to contention, differentiated services, or path changes) and to cross-traffic sharing a common bottleneck.

A. Late comer problem

Delay-based methods rely on measuring a base delay – the minimum delay of a specific network path. An incorrect measurement that assumes too low a delay can be detected. A higher measured base delay is more difficult to determine. This may often be a problem when a flow starts using an already congested bottleneck – including the queuing delay of other flows as a part of its measured delay. This leads to the 'late comer's advantage' problem. The late-comer flow thus pushes out the early flow and uses up the entire capacity. Work in [19] looked into this problem for LEDBAT flows.

We update our operational delay interval [d_{min} , d_{max}] using information from recent reports; so up-to-date data are used and occasional spurious measurements are quickly purged from the system. The use of an upper limit for d_{max} proportional to the size of the receiver playout buffer provides further resilience. Our algorithm is robust against the late comer's effect as shown in Fig.3, where the two video flows originate at different times, but none takes advantage of the other.

B. Route Changes

Path characteristics can change. Another problem that results from measured delay is when the sender reduces its rate in response to a reported delay increase caused by a route change, while there is no congestion. TCP Vegas suffered from this problem and a proposal was given by [20] in to solve it.

The proposed use of updated information means that our CDM learns the current delay characteristics following a path change. Faster adaptation can be obtained by observing packet jitter and by using packet-pair techniques. However, such methods have not been proven in the general Internet and we assume that path-changes are relatively infrequent and therefore are not a primary target.

C. Losses and Persistent Queues

Incorrect estimates of the path RTT may lead to delay-based methods contributing to persistent queues along a network path due to packets arriving at already full queues, leading to loss. In the case of packet losses, a congestion control method needs to significantly reduce the sender rate, to avoid further losses and potential congestion collapse.

The proposed algorithm promptly reacts to manifest congestion by halving the rate at every loss report. The delay associated to this loss event is used to update the value of d_{max} . Following the sharp reduction, a gradual rate increase is only pursued when the delay is consistently low. One way to improve the response time is to predict the future delay values by extrapolating current measured delays and observing the trend or their gradient. The effectiveness of such approaches in the Internet needs to be evaluated using data for actual network paths. We defer this validation to our future work.

D. Effects of Return Path Congestion

The return path contributes to a measured RTT. Congestion on the return path may therefore result in inaccurate estimates of the forward path characteristics causing a sender to reduce its rate even though the forward path is not congested. The effect of return path congestion is reduced in our system by using a one-way forward delay measurement instead of RTT.

If there is loss of feedback, the sender needs to assume that there may be excessive congestion (on either path) and needs to respond promptly to preserve network stability, but needs to be robust to occasional loss. We therefore assume that if no feedback arrives within two consecutive reporting intervals, a sender should reduce its rate linearly every two consecutive reporting intervals, until it receives a feedback updating the status of the network. A similar approach is proposed in [23].

VI. Coexistance with TCP and Other Traffic

Delay-based algorithms, including the delay-based control for the CDM proposed in [14], are inefficient when coexisting with loss-based methods like TCP. When a flow using a delay-based mechanism shares with TCP flows, it reduces its rate in response to increasing path delay (caused by TCP packets

filling the bottleneck queue). In turn, a TCP flow may continue to probe and fill any freed capacity, gaining a larger proportion of the bottleneck traffic.

TCP is not the only congestion control that can impact delay-based methods. In general, delay-based mechanisms yield to high load flows. This includes small flows (transferring small volumes of data that may not be congestion-controlled), and unresponsive flows (that do not implement congestion control or implement a method different to standard TCP).

The CDM presented in this paper overcomes the limitations of the pure delay-based approach of [14]. In essence, the proposed CDM adaptively switches amongst three available strategies to respond to different circumstances and path characteristics. The different and possibly complementary approaches to rate-control are governed by the following:

1. The media flow that shares a capacity bottleneck needs to offer low delay and responsibly share the available capacity with flows that use the same or similar forms of congestion control. This includes delay-controlled flows that have different path characteristics (RTT). Long-term coexistence with delay-controlled flows only is unlikely hence this approach alone does not suffice.
2. In most practical networks, capacity is also shared with other traffic. The congestion control algorithm therefore needs to accommodate other (background) traffic, such as web page viewing, DNS, etc. This competing traffic adds delay and induces losses that delay-controlled flows avoid. Overall this traffic is intermittent and typically a video flow experiences the delay it expects when adjusts its rate.
3. Multimedia streaming needs to be robust to persistent queue-building and loss-inducing traffic, such as bulk TCP flows. These flows will not readily yield to a delay-based algorithm. Furthermore they induce repeated loss events.

By default our CDM assumes to be competing with delay-controlled flows and employs the rate-control method described in section IV. Multiple flows, streamed with our system and unaware of each other, can compete for limited link capacity, discovering the bottleneck capacity and reacting to loss or delay by reducing their rate. Following a change of traffic, the flow converge to a fair share of the capacity; once this equilibrium has been reached the delay stabilizes, no loss occur, and the visual quality is steady, as seen in Fig.3. The same reaction, however, leads to a different result when the bottleneck is shared with other traffic such as TCP. The impact of such queue building and loss-inducing flows depends on the network bottleneck they experience:

- Small buffers: When the bottleneck buffer space is small, traffic will overflow faster and losses will be frequent, but the path delay will remain relatively short. The congestion control algorithm should be able to deal with these losses.
- Large buffers: If the bottleneck buffer is large compared to the time to serialise a packet, then large path delays may be observed and persist (with losses being infrequent).

Both cases are problematic, as a control scheme solely designed to avoid losses and delay will drive the sending rate down to zero and possibly terminate the flow. Large buffers particularly impact performance as the associated long delay may cause the video receiver to drain its playout buffer, causing the media playout to stall. In this case, the media client needs to increase the amount of buffered content before playing the video. In the following we focus on small to medium network buffers that do not imply this scenario.

The proposed CDM detects when delay and losses grow despite consistent efforts to reduce them. This is interpreted as a sign that (at least) one TCP-alike flow is sharing a bottleneck link. The CDM then switches to a different operational mode. In such TCP-aware modes the CDM adjusts the rate in a way that encourages TCP flows to reduce their capacity share.

The success of this method will depend on the number and arrival rate of the sharing TCP flows. In the case of many TCP flows, it is unlikely that any sender-based method will be able to reduce the path latency and in the absence of methods such as AQM or ECN, the most appropriate method appears to be a loss-based congestion control. We suggest two alternative congestion control approaches: *Safe Retreat* and *Responsible Aggression*.

Safe Retreat – By default the CDM starts using the delay-based approach to transmit 5 PIDs (50% of the nominal video rate). Delay and losses cause the CDM to progressively reduce the rate, as shown in Fig.4. If the video flow is consistently forced to stay at a minimum rate with reported packet losses, then the CDM switches to Safe Retreat mode. In this mode the CDM claims a portion of the capacity to stream, at low rate, the packets encoding the video at minimum quality. To combat the effect of loss we add resilience in form of packet FEC (e.g. PID1 +FEC for a rate of 20%; or PID1+FEC PID2 for a rate of 30%) as shown in Fig.4. The result is stable, albeit minimal, visual quality. This may continue as long as TCP probes the link causing losses to the video flow. If the flow cannot sustain the base rate (with FEC) then it is terminated. This "retreat" leaves ample room for TCP connections to achieve (more than) fair throughput. When the the TCP-induced losses cease and the delay rapidly dives, the CDM reverts to the delay-based approach: the FEC is dropped and the video flow gradually increases the streaming rate reclaiming any unused capacity.

On the other hand, if the delay remains high and moderate but frequent loss events are recorded over a certain interval, the CDM infers the presence of a long-lived TCP flow. In this case it switches to Responsible Aggression to claim a fair share of the capacity.

Responsible Aggression – In this mode the CDM increases the transmission rate from the minimum values of the Retreat (but not more than TFRC would). As with the previous mode,

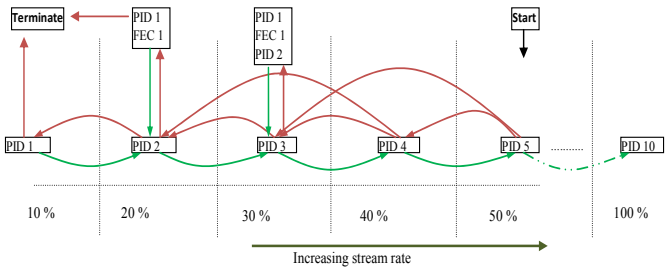

Fig. 4. Safe Retreat

FEC packets are used to protect the coded data. Packet losses are expected as the CDM tries to gain a fairer share of the link.

Starting from the Retreat value, the rate is increased by injecting data from one additional PID at a time, followed by copies of the same PID. If no loss is observed over a given interval, the process is repeated. In case of loss, the rate must be bounded by the throughput equation specified for TFRC (1) using the measured RTT, the loss fraction (in place of the loss event rate), and the average video packet size (e.g. obtained from the RTCP Receiver Reports).

$$\text{Transmit rate} = \frac{s}{R\sqrt{\frac{2pb}{3}} + \text{RTO}\left[3\sqrt{\frac{3bp}{8}}p(1+32p^2)\right]} \text{(Bytes/second)} \qquad (1)$$

where: s is the average packet size in bytes; R is the round trip time in seconds; p is the loss event rate (here replaced by the loss fraction); RTO is the equivalent TCP retransmission time in seconds, approximated by RTO=4R; b is the number of packets that would be acknowledged by a single TCP acknowledgement, approximated by $b = 1$.

Overall this mode implements a loss-based rate control where rate fluctuations are smoothed as much as possible. The system tries to find equilibrium with TCP to deliver a stable, albeit reduced, visual quality. If link capacity suffices, the flow reaches the nominal video rate, although this time the stream includes the FEC packets within this amount, hence the visual quality is lower than the nominal one, losses notwithstanding. If the link is persistently overloaded rate is reduced till returning to Retreat. Conversely, consistent delay reduction and no loss trigger the switch to delay-based rate control.

A. Experimental Results and Discussion

We evaluated our proposal by simulating the network in Fig.2 where a single 700 kbps video flow shares a 1.5 Mbps bottleneck with a TCP (FTP) flow. Fig.5 shows the sending rate. The CDM start operating in delay-based mode. In the first 100 s, the video has the entire capacity to itself, delay is very low, and the CDM it increases the rate steadily. Before it achieves the nominal rate, the TCP flow starts and quickly floods the 600ms buffer causing losses. The loss triggers a reduction in rate, first from 8 PIDs to 4 PIDs, then in the following two intervals down to the PID 1 (base layer).

During the interval following the sharp rate reduction (100-150s), the algorithm senses that it may be sharing capacity with a TCP flow because even after driving the video rate to the base layer, the delay still remains high enough to hinder it from growing back its rate. It therefore goes into the Safe Retreat mode (as a transient step into the 'responsible aggression' mode) by adding PID 2 and also FEC packets of PIDs 1 & 2. This allows the video to claim some capacity equivalent to 4 PIDs of rate while protecting the base layer. Since TCP still probes for capacity, the delay persists with some losses for more than three consecutive reporting intervals. As a result, the algorithm enters the Responsible Aggression mode where it gradually increases the video rate, as described previously, till loss occurs and TCP backs-off. This allows the video flow to claim some share of the capacity. As the video flow share of the capacity grows, further increases become less likely, due to

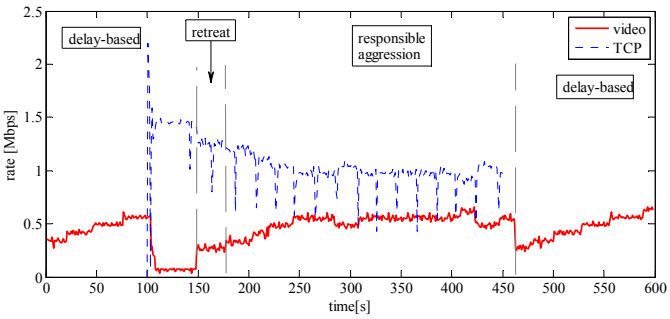

Fig. 5. Video and TCP send rates

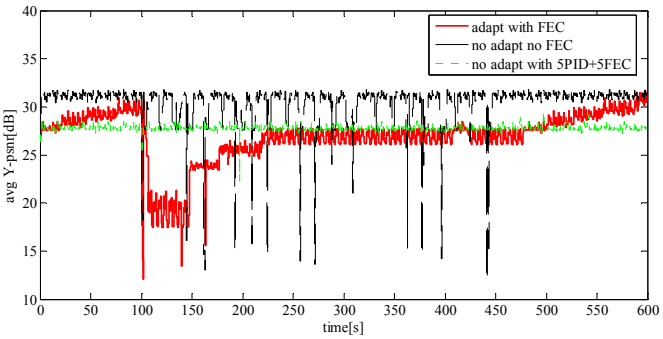

Fig. 6. Average PSNR of received video after sharing capacity with TCP

the CDM congestion control based on (1). The capacity share claimed in the example of Fig.5 seems at least fair to TCP. When the TCP flow completes transmission, the CDM detects the conditions to switch to the delay-based rate control: the video drops the no longer required FEC packets (rate drop at 470s) and progressively returns to nominal rate.

The receiver's video PSNR is shown in Fig.6, with the thick solid line corresponding to the video flow of Fig.5. In order to quantify the loss in quality resulting from TCP interference we report the performance of two reference video flows, each independently streamed under the same network conditions and with the same TCP flow of Fig.5. The two reference flows, constantly streaming at nominal rate without any congestion control, are: the flow containing the full video content (dashed dark line Fig.6) and the flow containing 50% of the content and 50% FEC protection (dotted green line Fig.6). The first reference provides an upper bound for the quality achievable (in principle) during the intervals when no loss occurs; it also demonstrates that when losses do occur, the quality drop is sudden and obvious. The second reference shows the quality that could be achieved, using FEC to protect against possible losses, if one knew a-priori the fair share of the capacity.

As shown in Fig.6, during the interval 100-150s coexistence with TCP drives down the rate and induces losses, leading to a quality around 20 dB. Once our system switches to TCP-aware modes, the quality improves, climbing up to 27 dB before the rate is limited by TFRC as in (1). The maximum quality it achieves is equivalent to 4 PIDs. The first reference, dashed line, shows brief but severe quality fluctuations when the video flow shares with TCP, because base layer packets may be lost whenever TCP overflows the buffer (similar but

smaller fluctuations are observed also in the adaptive flow in the interval 100-150s). This type of fluctuation is not desirable for the end user. The second reference, dotted line, shows a consistent PSNR corresponding to 5 PIDs (almost always delivered due to FEC, as the chances of losing both a PID and its corresponding FEC are slim).

The result of Fig.5 is evidence of the merit of the proposed system. It confirms that it can detect the presence of a TCP flow and then coexist with it. The comparison with the (second) reference system in Fig.6 shows that our system reaches almost to the same quality as the reference system that has a-priory knowledge of the fair share of the rate (without incurring the quality fluctuations of the other reference system).

VII. Conclusion

Interactive network multimedia has expectations that differ significantly from the properties of transport flows derived on TCP. This paper explores methods that are designed to minimize the impact of loss and avoid excessive delay. The approach discusses these methods related to a framework, asserting that solutions that consider only transport or media adaptation are suboptimal and a cross-layer optimization is essential.

Our framework utilizes a scalable video codec. This has two primary advantages: First it allows a flexible and low-cost adaptation of the media rate driven by a congestion control transport protocol. Second the scalable video format allows additional FEC to be introduced to protect key parts of the media flow from the effects of congestion-induced loss. This can make informed decisions on what media to send to optimize video quality to available network capacity.

Proactive congestion control mechanisms are essential to maintain acceptable interactive multimedia performance. However we also conclude that there are fundamental limitations to methods that rely only on delay measurements as congestion signals. Instead we assert that delay has to form one important metric but must for robustness be combined with methods to promote fair coexistence with other flows sharing the capacity of a network bottleneck.

Acknowledgment

This research was partially funded by the European Community under its Seventh Framework Programme through the Reducing Internet Transport Latency (RITE) project (ICT-317700). The views expressed are solely those of the author(s).

References

[1] IETF RMCAT WG Charter, Available at: http://datatracker.ietf.org/wg/rmcat/.

[2] M. Allman, V. Paxson and E. Blanton, "TCP Congestion Control," *IETF RFC 5681,* September, 2009.

[3] S. Floyd, M. Handley, J. Padhye and J. Widmer, "TCP friendly rate control: TFRC Protocol specification " *IETF RFC 5348,* September, 2008.

[4] ISO/IEC JTC 1/SC 29, " MPEG-2 (Generic coding of moving pictures and associated audio information)," *ISO/IEC JTC 1/SC 29,* November, 2009.

[5] ITU-T Recommendation H.264, "Advanced video coding for generic audiovisual services," *SERIES H: AUDIOVISUAL AND MULTIMEDIA SYSTEMS,* May, 2003.

[6] L. S. Brakmo, S. W. O'Malley and L. L. Peterson, "TCP Vegas: New techniques for congestion detection and avoidance," in *Proceedings of the Conference on Communications Architectures, Protocols and Applications SIGCOMM,* New York, 1994, pp. 24-35.

[7] J. Martin, A. Nilsson, I. Rhee, "Delay-Based Congestion Avoidance for TCP", *IEEE/ACM Transactions on Networking*, vol. 11, Issue 3, pp. 356-369, 2003.

[8] S. Shalunov, G. Hazel, J. Iyengar and M. Kuehlewind, "Low Extra Delay Background Transport (LEDBAT)," *IETF RFC 6817,* December, 2012.

[9] K. Winstein, A. Sivaraman, H. Balakrishnan, "Stochastic Forecasts Achieve High Throughput and Low Delay over cellular Networks", *USENIX Symposium on Networked Systems Design and Implementation,* Lombard, IL, April 2013.

[10] K. Ramakrishnan, S. Floyd and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *IETF RFC 3168,* September, 2001.

[11] R. Jain, "A delay-based approach for congestion avoidance in interconnected heterogeneous computer networks", *Comp. Comm*. Rev. 9(5), pp. 56-71 (1989).

[12] K. Tan, K. Song, Q. Zhang, M. Sridharan, "A Compound TCP Approach for High Speed and Long Distance Networks", *IEEE INFOCOM '06*, pp. 1-12.

[13] L. Budzisz, "On Fair coexistence of Loss- and Delay-based TCP", *IEEE/ACM Trans. On Networking*, Vol. 19, Issue 6, 2011, pp. 1811-1824.

[14] N. Iya, F. Verdicchio, R. Secchi, G. Fairhurst, "Rate Adaptation and Congestion Avoidance for Scalable Video Streaming", *14th Annual PGNet Symposium,* Liverpool, June 2013.

[15] H. Schwarz, D. Marpe and T. Wiegand, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 17, pp. 1103-1120, 2007

[16] Do-Kyoung Kwon, Mei-Yin Shen and C. -. J. Kuo, "Rate Control for H.264 Video With Enhanced Rate and Distortion Models," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 17, pp. 517-529, 2007.

[17] P. Lambert, W. De Neve, P. De Neve, I. Moerman, P. Demeester and R. Van de Walle, "Rate-distortion performance of H.264/AVC compared to state-of-the-art video codecs," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 16, pp. 134-140, 2006.

[18] Jin Yang, Yu Sun, C. S. Kline and Shixin Sun, "Adaptive initial quantization parameter selection for H.264/SVC rate control," in *Intelligent Computing and Intelligent Systems (ICIS), 2010 IEEE International Conference on,* 2010, pp. 723-726.

[19] G. Carofiglo, L. Muscariello, D. Rossi, S. Valenti, "The Quest for LEDBAT Fairness", *IEEE GLOBECOM 2010,* pp. 1-6.

[20] K. N. Srijith, L. Jacob, A. L. Ananda, "TCP Vegas-A: Solving the fairness and rerouting issues of TCP Vegas", *Proceedings of the IEEE International Performance, Computing and Communications Conference,* 2003, pp. 309-316.

[21] "The Network Simulator – ns-2". http://www.isi.edu/nsnam/index.php/User_Information, November 5, 2011.

[22] J. Iyengar, B. Ford, D. Ailawadi, S. O. Amin, M. Nowlan, N. Tiwari, J. Wise, "Minion—an All-Terrain Packet Packhorse to Jump-Start Stalled Internet Transports" PFLDNet 2010, November 2010.

[23] C. Perkins and V. Singh, "RTP Congestion Control: Circuit Breakers for Unicast Sessions draft-perkins-avtcore-rtp-circuit-breakers" *IETF Internet Draft,* February, 2013.