Exploration of RTP circuit breaker with applications to video streaming.

FOUGH, N., VERDICCHIO, F. and FAIRHURST, G.

2013

Exploration of RTP Circuit Breaker with Applications to Video Streaming

Nazila Fough, Fabio Verdicchio, Gorry Fairhurst
Electronic Research Group, University of Aberdeen, Aberdeen, UK
{nazila.fough, fverdicc, g.fairhurst}@abdn.ac.uk

Abstract—**Live multimedia streaming is becoming one of the dominant sources of internet traffic, much of which is sent over best-effort networks, i.e. along paths with a wide variety of characteristics. The multimedia traffic should be transmitted using a robust and effective congestion control mechanism to protect the network from congestion collapse. The RTP Circuit Breaker (RTP-CB) is a candidate solution that causes a sender to cease transmission when RTCP message feedback indicates excessive congestion. This paper studies RTP/UDP video traffic and the impact of its bursty behavior on the network. It considers the potential limitations of using a RTP-CB with video traffic. We found that the bursty nature of a typical video flow can cause the RTP-CB to either *prematurely* cease transmission or *to react too late*. To reduce the likelihood of this happening, we suggest the use of a smoothing buffer in conjunction with the RTP-CB and propose design criteria for this buffer. Our experiments confirm the effectiveness of the proposed approach for different video streams.**

Index Terms—RTP/UDP video traffic, RTP-Circuit Breaker

I. Introduction

Multimedia streaming, video conferencing and other real-time multimedia applications are becoming increasingly popular and are expected to dominate Internet traffic in the near future. Deployment of applications using the RTCWEB protocol [1], and expected increase in media streaming pose considerable challenges for sharing the available capacity with other traffic in existing wired/ wireless network infrastructure. To prevent network congestion, there is an urgent need for these methods to support some form of congestion control. However, at the time of writing this paper, there is no generally accepted congestion-control method for these media flows [2].

A TCP flow adapts its rate based on a congestion control algorithm utilising feedback received over the network. Hence a controlled flow adapts its sending rate, by taking into consideration the state of the network path, and reacts to avoid impending congestion. Internet video traffic is usually carried using RTP/UDP, which does not implement congestion control at the transport layer, with the rate simply determined by the video codec settings. Hence uncontrolled UDP video traffic has the potential to induce significant congestion. Furthermore, when controlled (e.g. TCP [3]) and uncontrolled (e.g. UDP [4]) multimedia flows share a link, the well behaved flows reduce their rate and capacity usage, while the unrestricted ones unfairly dominate the bottleneck usage and retain the potential to congest the bottleneck, thus damaging all the flows.

The RTP Circuit Breaker (RTP-CB), recently proposed in an Internet Engineering Task Force (IETF) internet-draft [5], is a technology that is expected to protect the network from excessive congestion. A media sender that receives notification of congestion should invoke the RTP-CB to cease transmission to avoid further congestion. It is expected that well-controlled RTP applications using best-effort networks will be able to operate without triggering the RTP-CB, while misbehaving flows will be blocked to prevent congestion and avoid damaging other users.

In this paper we review the relevant features of a video streaming session (Section II). We link these features to the effective use of the RTP-CB as a safety switch for a video flow (Section III). We highlight potential problems with the RTP-CB and propose a solution based on a pacing buffer (Section IV). To our knowledge this study has not been performed before. We provide preliminary experimental evidence that support the effectiveness of the proposed scheme (Section V) and discuss our findings (Section VI).

II. Video Streams over Networks

We begin by reporting the transmission profile of a typical video stream. The sample video sequence is denoted as Test1, it is one of the sequences considered in this paper and described in Table 1 (Section V). The streaming system architecture comprises a streaming server (acting as media provider) a network emulator (simulating a bottleneck link with limited capacity and a drop-tail buffer) and a video client (representing the user). The testbed is further described in Section V. The transmission profile for a typical eight-second segment of the sequence is shown in Figure 1. Each point in the figure represents the average transmission rate over a one second interval. The sequence is encoded at a nominal rate of 1 Mb/s and the points in the graph lay around this value. For the same video segment, Figure 2 shows the *instantaneous* transmission profile: each point in the figure represents the average transmission rate over a short interval of 5 ms. The figure shows the presence of distinctive spikes reaching instantaneous rate more than ten times above the nominal rate. We report that these spikes are a characteristic of every video streaming session we considered.

Streaming over a limited capacity link following the transmission profile of Figure 2 can severely impact the quality of the received video. For example, we streamed the video sample Test1 over a path with a capacity of 1.5 Mb/s and a bottleneck buffer size of approximately 10 packets. The link capacity is greater than the nominal rate of the video (1 Mb/s) hence the moderate size of the buffer, in line with current trends [6], would seem appropriate. We would therefore expect

The work of N. Fough is supported by EPSRC.

flawless reception of the video as the path is not shared with any other flow. The results show that this is not the case. First, a moderate loss rate, reported in Figure 3, is observed. For the segment reported in the figure, the average loss rate is less than 6%. Second, the decoded visual quality is remarkably poor, as demonstrated by the sample frame shown in Figure 4. Such low quality is unexpected for a low loss rate, hence further explanation is necessary.

A video stream consists of a succession of Groups of Pictures (GOPs), each comprising a series of coded frames. The frames in a GOP can be grouped in two categories:

- Intra-coded frames: Known as *I-Frames*. Any I-Frame is coded independently of other frames and typically spans several packets, all of which must be received to ensure correct reconstruction of the frame in the decoded video.
- Predicted frames: These are encoded using previously transmitted frames and are very efficiently compressed, requiring far fewer packets than I-Frames [7].

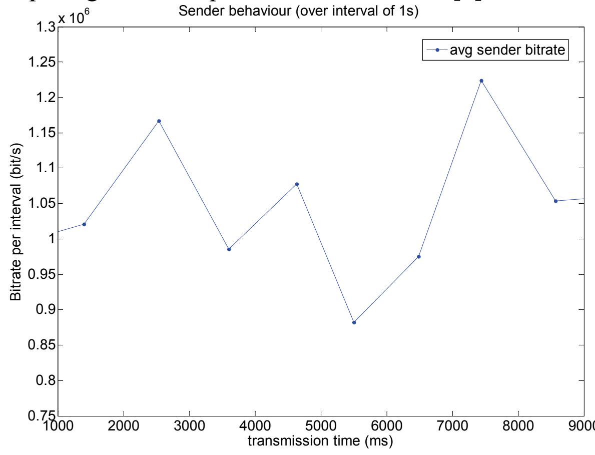

Figure 1. Transmission profile for a segment (seconds 1-9) of video Test1. Each point represents the average rate over 1 s.

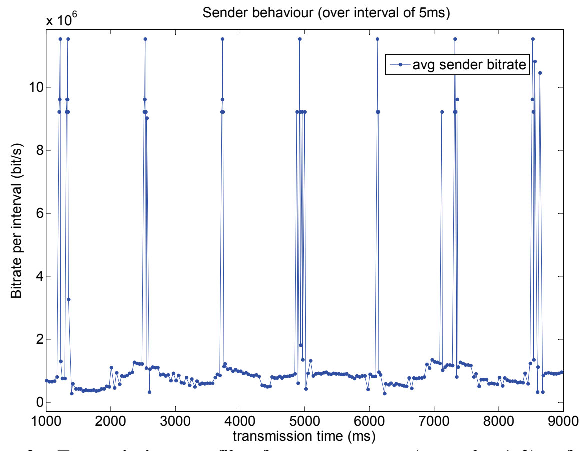

Figure 2. Transmission profile for a segment (seconds 1-9) of video Test1. Each point represents the average rate over 5 ms.

The typical GOP structure comprises one initial I-Frame (requiring many packets) followed by several predicted-frames (each requiring few packets). Increasing the number of predicted frames increases the compression ratio of the sequence. However, packet losses affecting an I-Frame lead to decoding errors that propagate to all the subsequent predicted frames. The GOP size is chosen when the sequence is encoded (typically prior to transmission and without knowledge of the network conditions). A trade-off value is set with the aim to retain coding efficiency while limiting the propagation of errors in the decoded sequence. A GOP size of 30 frames, corresponding to 1.2 seconds of video, was chosen for the streams used in this paper.

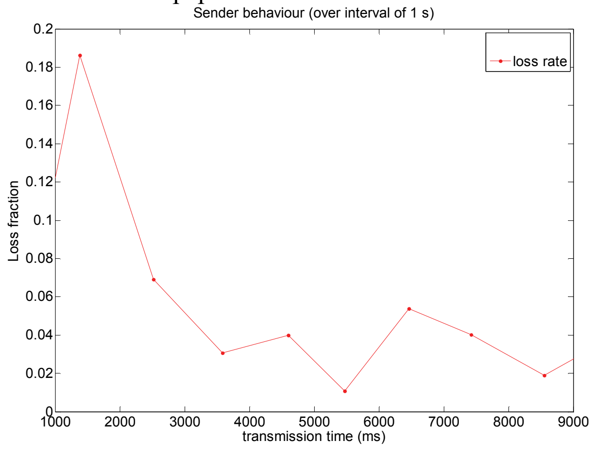

Figure 3. Loss rate for a segment (seconds 1-9) of video Test1. Each point represents the loss fraction over a 1 s interval.

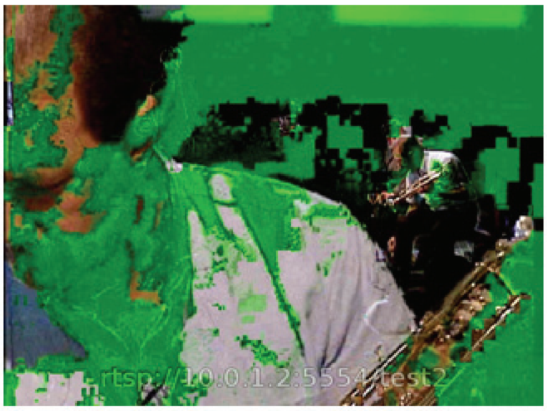

Figure 4. Sample decoded frame obtained streaming Test1 over a network path with capacity 1.5 Mb/s and buffer size of 10 packets.

In light of the above description, the transmission profile shown in Figure 2 is readily explained. First we show in Figure 5 the number of packets (payload 1440 bytes) required to transmit the frames of Test1. The pattern of repeating GOPs is clearly visible in the figure, with each GOP comprising 30 frames, one I-Frame (spanning several packets) followed by predicted frames (each spanning fewer packets). Every frame is transmitted over a 40 ms interval. Clearly, the transmission of an I-Frame over this interval requires a much larger rate than in the case of a predicted-frame. As a result, the instantaneous rate peaks when an I-Frame is transmitted, as confirmed by comparing Figure 5 with Figure 2. This behaviour is typical of the entire sequence. Analyzing the distribution of frame sizes for the Test1 sequence we found that the predicted-frames have sized of 1-10 packets (average 3.6), whereas the I-Frames are distributed in the 11-44 packets range (average 20). The underlining features of these statistics are not unique to this sequence. These results can be used to model the distribution of frame sizes, hence the spikes in the transmission rate, for different sequences. The use of this model is the subject of Section IV.

We conclude by explaining that poor video quality (Figure 4) is the inevitable outcome of the transmission profile in Figure 2, despite the modest loss rate reported in Figure 3. We compare the instantaneous loss rate, given in Figure 6, with the transmission time of the video frames, given in Figure 5. The transmission of I-frames incurs several losses, while the subsequent predicted frames are nearly unaffected. This results in poor-quality reconstruction of the affected I-Frames, which in turn significantly lowers the reconstruction quality of subsequent predicted-frames, although these may not be missing any packet.

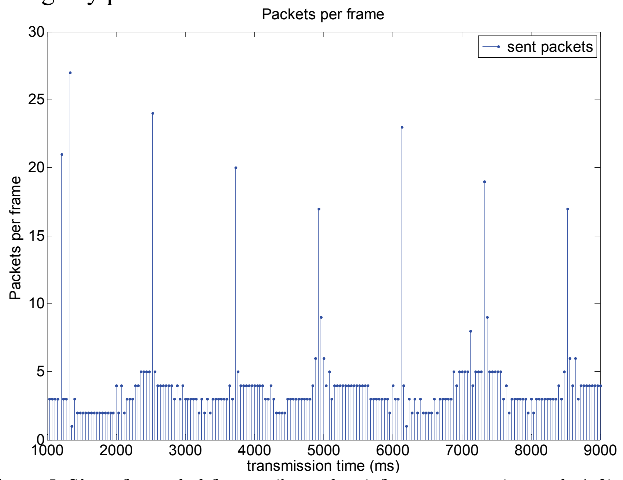

Figure 5. Size of encoded frames (in packets) for a segment (seconds 1-9) of video Test1. the pattern of one I-frame (that spans more than 15 packets) followed by several predicted-frames (each spanning less than 10 packets).

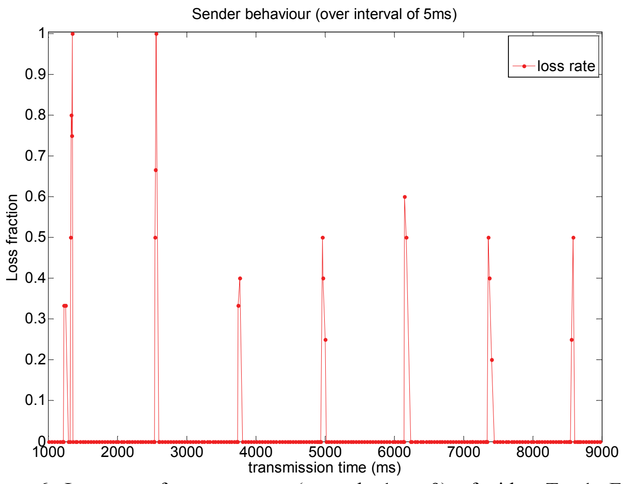

Figure 6. Loss rate for a segment (seconds 1 to 9) of video Test1. Each point represents the loss fraction over a 5 ms interval.

III. RTP CIRCUIT BREAKER (RTP-CB)

The RTP-CB, defined in [5] within the scope of the RTP/Audio Video Profile (AVP) [11], is not intended to be used as a rate-adaptation algorithm. Instead, it is a lightweight technology that specifies a minimal set of conditions under which an RTP flow should cease transmission to protect other flows sharing the network from excessive congestion. It is expected that RTP applications featuring rate control algorithms will operate normally without triggering the circuit breakers [5], while uncontrolled flows will trigger the RTP-CB. The objective is to achieve a fair coexistence of TCP, UDP and other traffic.

In a RTP-based streaming session, RTP-CB is co-located with the server and monitors the ongoing transmission. The RTP-CB should be able to detect one congestion-inducing RTP flow (i.e. one causing packet losses and delay in the network) using only basic RTCP features [12], without needing rapid RTCP reports. This paper considers *congestion* as the condition to trigger RTP-CB [5]: In the event of packet losses, an RTP flow, competing with other flows, should not be allowed to achieve an average throughput (measured on a reasonable time scale) larger than other flows (e.g. TCP). The following mechanisms are proposed to detect network congestion [5]: (*i*) estimation of Round Trip Time (RTT); (*ii*) calculation of the jitter (statistical variance of RTT) over the RTCP interval report; (*iii*) reception of a Receiver Report (RR) that signals packet losses. The first two signalling methods are useful *early-warning* of potential congestion, but on their own provide an insufficiently strong signal to be used by RTP-CB [5]. We therefore focus on packet loss, reported to the sender (and to the RTP-CB) via a RR every RTCP reporting interval, approximately once every 5 seconds. When losses are detected the RTP-CB should determine whether the (video) flow being monitored is overloading the link. This decision is made by comparing the transmission rate with the rate of an equivalent TCP flow experiencing the same RTT and loss rate. This rate can be estimated as [5]:

$$\text{Bit-rate} = \frac{\text{packet\_size}}{RTT \times \sqrt{\frac{2}{3}\,\text{loss\_fraction}}} \tag{1}$$

where the loss fraction approximates the TCP *loss-event rate* as explained in [5].

If the video flow rate is above the bit-rate given by (1), then the flow is deemed to be taking unfair advantage over a possible TCP competitor and the RTP-CB should disrupt this flow. Section II has shown that streaming a video trough an unshared link, with capacity almost 50% above the nominal video rate, can lead to periodic packet losses. In Section II we have shown that these losses occur whenever an I-Frame (spanning tens of packets) is transmitted over the typical 40 ms interval. Loss-free transmission would require a network capacity ten times higher than the nominal video rate (see Figure 2). Alternatively it would require large buffers, which may not be desirable in a real network as it would also increase delay. The RTP-CB is notified loss events via RRs. The RTP-CB will then compare the video rate with the equivalent TCP flow. On the one hand, the transmission rate may be found too high. If the application does not reduce the rate significantly (by at least a factor of ten) to resolve congestion, the video is terminated within two further RTCP reporting intervals [5]. For a video stream encoded at a fixed nominal rate (the case of the MPEG4 streams used in our experiments), live streaming cannot occur at a lower rate. Even worse, the bit-rate periodically surges (as in Figure 2). Hence the video flow will be terminated by the RTP-CB at the first opportunity. Any further attempt to stream the video will meet the same fate. While this outcome is in line with the intended RTP-CB behaviour, the video can be streamed at no risk for other link users if the rate is kept smooth (as in Figure 1) and losses therefore avoided. On the other hand, the rate may be found to be fair to the equivalent TCP flow and the video flow is then allowed to continue. Despite a non-zero loss-rate, the limited availability of path capacity and buffer size implies that a fraction of the packets encoding many I-Frames will be lost. As shown in Section II, this leads to reconstructing a poor-quality video. An impaired video flow thus could be allowed to continue, potentially causing packet loss to other flows sharing the link, while the video user is not notified of excessive congestion but experiences inexplicably poor video quality, as shown in Figure 4.

IV. RTP-CB with Pacing Buffer

The video streaming session described in Section II is monitored by the RTP-CB described in Section III. Considering the bit-rate, loss-rate and delay profile of video Test1 over five seconds (i.e. the RR interval using AVP [11]), the video rate is more than ten times above the TCP equivalent rate (1). Therefore the RTP-CB will terminate the video flow unless the application significantly reduces its rate. Rate reduction is not a simple task when the video is pre-encoded using MPEG4 [7]. The RTP-CB complies with its network-safety specification and ceases the video flow, despite the fact that the video rate, over a five-second timescale, is compatible with the path capacity. If the video rate cannot be lowered, the route to meet the requirements of (1) is to reduce or eliminate the packet losses. The analysis of Section II provides guidance. The short-lived loss events of Figure 6 arise whenever (several) packets comprising an I-Frame are injected into the network path in a burst (Figure 5). The key to reducing losses is smoothing these bursts. A pacing buffer, placed before the RTP-CB (Figure 7), is proposed to smooth the instantaneous rate of Figure 2 and render it similar to the well-behaved average rate of Figure 1.

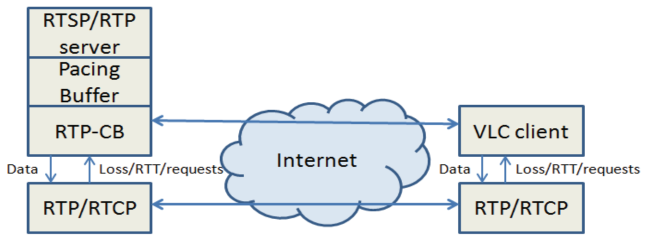

Figure 7. Client/server scheme with pacing buffer.

The pacing buffer (PB) is fed every Δ_T seconds with the packets encoding one video frame (in our examples Δ_T = 40 ms). The packets held in the PB are transmitted at a maximum output rate $R_{PB}^{\max}$. We label δ_T the time required for the PB to send a (regular-sized) packet, this is calculated as $\delta_T = 8\times\text{packet\_size}/R_{PB}^{\max}$. The maximum number of packets leaving the PB between the arrival of two frames is $K_{PB}^{\max} = \lfloor \Delta_T/\delta_T \rfloor$; we assume that $\Delta_T = K_{PB}^{\max}\delta_T$. Therefore the number of packets in the PB varies at discrete set of instants $t = n\delta_T$, with $n = 0,1,2,\ldots$, increasing following the arrival of a frame packet(s) or decreasing one packet at a time. We define the sequence $z[n]$ as the number of packets held in the PB at time $t = n\delta_T$. $z[n]$ is given by:

$$z[n] = z[n-1] + x[n] - y[n] \tag{2}$$

where $x[n]$ denotes the number of packets fed to the PB at time $n\delta_T$, and $y[n]$ denotes the number of packets leaving the PB at time $n\delta_T$. Initially the PB is empty ($z[-1]=0$). The PB input $x[n]$ is a stochastic process modelling the number of packets encoding a frame when this is fed to the PB, i.e. when $n = k\cdot K_{PB}^{\max}$ (with $k = 0,1,2,\ldots$), and zero otherwise, i.e. $x[n]=0$ when $n \neq k\cdot K_{PB}^{\max}$. The statistical analysis of $x[n]$ is beyond the scope of this paper. The following simplified representation suffices:

$$x[n] = \begin{cases} N_I & \text{if} \quad n = k\cdot\text{GOP\_size}\cdot K_{PB}^{\max} \\ N_P & \text{if} \quad n = (k\cdot\text{GOP\_size} + j)\cdot K_{PB}^{\max} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where N_I and N_P denote the *average* number of packets that make up an I-Frame and a Predicted frame respectively, $j = 1,2,\ldots\text{GOP\_size-1}$. For our examples, GOP comprising one I-Frame followed by 29 predicted-frames, $x[n]$ is shown in Figure 8 (top). Finally, $y[n]=1$ when the PB is non-empty, $y[n]=0$ otherwise. The PB intended behaviour required that PB can transmit, for each GOP, at least as many packets as it receives. For this to happen, the PB output rate should be:

$$R_{PB}^{\max} = \frac{N_I + N_P(\text{GOP\_size}-1)}{\text{GOP\_size}} \frac{8\times\text{packet\_size}}{\Delta_T}. \tag{4}$$

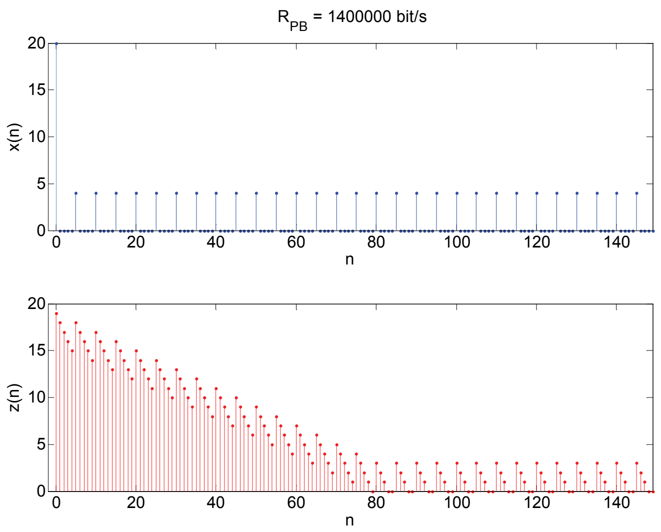

Figure 8: Top: Pacing Buffer input packets (3) for Test1. Bottom: buffered packets (2) for Test1. This pattern corresponds to a typical GOP (duration 1.2 seconds) and repeats periodically.

The value of $R_{PB}^{\max}$ must be at least that of (4) to ensure that all packets encoding one GOP are transmitted before the next

GOP enters the PB. In this case, as shown in Figure 8 (bottom), the PB size does not exceed the size of a coded I-Frame. Setting $R_{PB}^{\max}$ below (4) would lead to (linear) growth of the PB size with time. Assuming the streaming scenario of Sections II and III, we set $R_{PB}^{\max}$ =1.4 Mbit/s, slightly above the value given by (4). We observe that the loss events of Figure 6 disappear completely and the RTP-CB is not triggered. Conversely, if the bottleneck-link cannot sustain this video rate, losses systematically occur and the RTP-CB duly triggers. With the addition of the PB, the RTP-CB now manages the video flow effectively.

The transmission of regularly spaced video packets avoids congesting the bottleneck, but packets accumulate delay in the PB rather than in network buffers. This delay, however, is predictable for a typical GOP and is balanced by the client using a *play-out buffer*. Video clients usually buffer incoming packets before starting the decoding process. When the streaming rate matches the link capacity, the play-out buffer receives packets encoding one GOP at the same rate at which the decoder retrieves them to decode and display the video. However, as all packets are equally paced, it takes longer to receive the several packets of an I-Frame than the few packets of a predicted-frame. We require that *all* packets comprising an I-Frame are in the play-out buffer when the decoding of this frame begins. The PB model described above allows determining the minimum size for the play-out buffer. For one GOP this corresponds to buffering the N_I packets of the I-Frame. Furthermore, we need to avoid that the play-out buffer depletes when the I-Frame of the successive GOP is to be played: enough predicted-frames packets must be stored, before decoding begins. This ensures that the play-out buffer never depletes at GOP boundaries. The resulting decoding start-up delay is:

$$\text{startup\_delay} = N_I\, \delta_T \left(1 + \frac{N_P}{K_{PB}^{\max}}\right) \simeq N_I \frac{16 \times \text{packet\_size}}{R_{PB}^{\max}} . \quad (5)$$

The start-up delay (5) can be reduce by increasing the PB rate $R_{PB}^{\max}$. However $R_{PB}^{\max}$ should not exceed the available path capacity. For the streaming of Test1, with $R_{PB}^{\max}$=1.4 Mbit/s, the delay (5) is 340 ms. We note that (5) uses average values for N_I. Therefore this start-up delay may not be sufficient to cope with a larger-than-average I-Frame and the decoding would then freeze till enough packets are in the buffer. The likelihood of this event is reduced if (5) is slightly overestimated. We use 400 ms in our experiments.

V. EXPERIMENTAL RESULTS

Our testbed comprises:

- A streaming server (RTSP/RTP-LIVE555 [9][10] using Debian 2.30.2) acting as media provider.
- A VLC client (Ubuntu 11.10) represents the user and was used to provide subjective quality assessment.
- A network emulator (Netem packages on Ubuntu 11.10 [8]) to simulate a network bottleneck (a token bucket filter and Netem at the LAN interface limit the rate and provide a fixed-sized drop-tail buffer).
- The sequences used in our experiments, listed in Table 1, are selected from those produced in the SUNRISE Project [13]. All videos have resolution 720x576 and are encoded with MPEG4 at 25 frames per second.

Table 1. Video sequences used in our streaming experiments.

Label	Content Name	Rate (Kb/s)	Duration	N_I	N_P
Test1	Mutes1	1200	00:49	20.2	3.6
Test2	Mutes1	2400	00:49	46.6	6.3
Test3	Star Wars	500	00:33	11.8	1.5
Test4	Star Wars	1200	00:33	19.6	3.4
Test5	Star Wars	2400	00:33	38.5	6.6
Test6	Star Wars	4400	00:33	59.1	13.6

We begin by showing the streaming profile obtained when the same video content considered in Section II is encoded at (approximately) double the nominal rate. The instantaneous bit-rate requirement of this flow (Test2) is shown in Figure 9. Without the use of the pacing puffer the bit-rate profile is characterized by periodic spikes, reaching an order of magnitude above the nominal rate. These are similar to spikes observed in Figure 2 (the spikes in Figure 9 are wider than those in Figure 2). As in Section II, the rate peaks match the pattern of coded frames. The frame-size distribution for the Test2 sequence is consistent with the one reported in Section II for Test1: the average sizes for predicted frames and I-Frames are approximately double the values observed for Test1. This trend, i.e. approximately doubling of the frame size with doubling of the rate, is observed for all the sequences reported in Table 1. Analogously to the discussion of Section III, streaming Test2 over a path with 3 Mbit/s capacity (i.e. above the nominal video rate) incurs loss bursts that trigger the RTP-CB. The pacing buffer is used to smooth the streaming rate profile. The resulting profile is also shown in Figure 9.The PB rate $R_{PB}^{\max}$ given by (4) matches the path capacity, losses are avoided and the RTP-CB is not required to take action. The analysis of Section IV allows calculating the start-up delay (5) associated with the play-out buffer at client side. For Test2 we keep the start-up delay assumed for Test1 (400 ms), as both $R_{PB}^{\max}$ and N_I in (5) are almost doubled. Our experiments confirm that the play-out buffer never depletes.

Similar results are obtained for the other sequences listed in Table 1. Two examples are given in Figure 10 and Figure 11 for sequences Test 4 and Test 5 respectively. Without pacing buffer, the streaming rate periodically peaks, following the repeating GOP pattern. For the network settings assumed in this paper, i.e. path capacity slightly above the nominal video rate and moderate router buffer sizes, all streaming sessions suffer periodic packet losses. Albeit of limited duration, these events lead to poor-quality video reconstruction (as in Figure 4) and typically violate the bit-rate limit given by (1), causing the RTP Circuit Breaker to terminate the flow.

The usage of the pacing buffer constrains the bit-rate profile to be below $R_{PB}^{\max}$ at all times, as shown in the figures. $R_{PB}^{\max}$ is set to equal the nominal video rate. When this matches the path capacity, losses are avoided and the RTP-CB is not triggered. If the nominal video rate cannot be sustained by the path, the

RTP-CB appropriately terminates the flow. While promising, these results are still preliminary and further experiments, involving a wider set of sequences and additional network settings, are under-way.

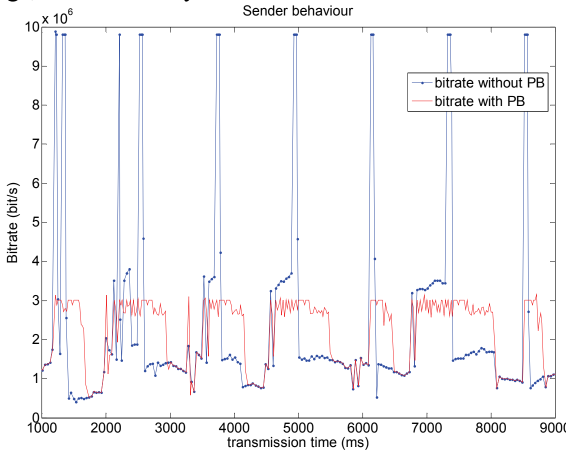

Figure 9. Transmission profile for a segment (seconds 1-9) of video Test2 with and without pacing buffer.

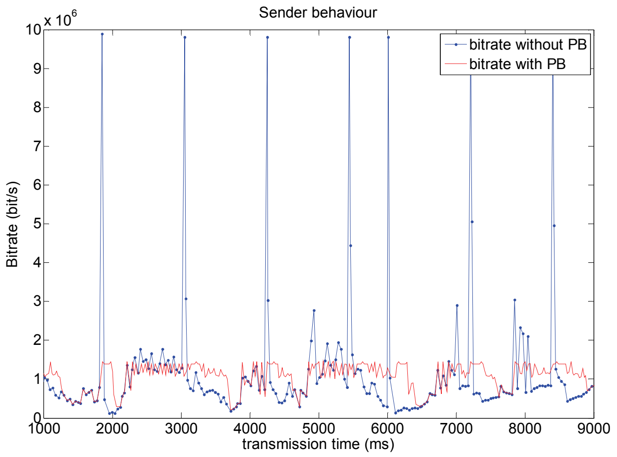

Figure 10. Transmission profile for a segment (seconds 1-9) of video Test4 with and without pacing buffer.

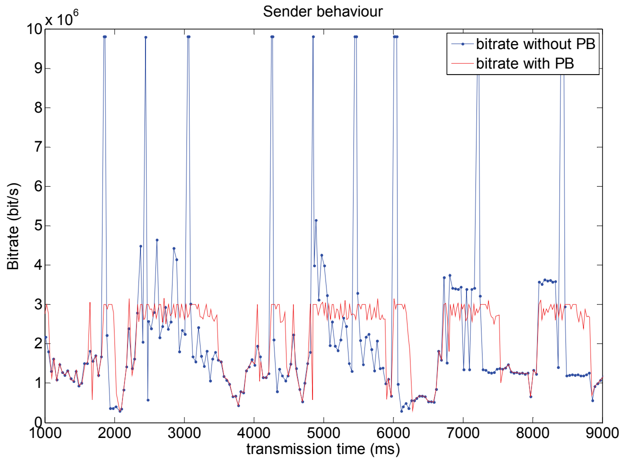

Figure 11. Transmission profile for a segment (seconds 1-9) of video Test5 with and without pacing buffer.

VI. Conclusion & Discussion

This paper explores the effectiveness of the recently proposed RTP Circuit Breaker (RTP-CB) when used with high-quality variable-rate video sent over RTP/UDP. A flow that is irresponsive to loss events will cause TCP traffic to decrease its rate and enable UDP traffic to take more than its fair share of capacity. The role of the RTP-CB is to control the aggressive behaviour of a RTP/UDP video stream and terminate an irresponsive flow. We observed that video streams can become unusable as a consequence of moderated losses experienced even when links are not apparently over utilized. To understand the root causes of this effect, the video characteristic was studied. The results show that video streaming sessions generate bursts of packets that overload network bottleneck with moderate buffers. For many applications, increasing the size of network buffer is not an option, so we studied the reaction of the RTP-CB to a video flow experiencing bursty losses. We observed a high loss rate in the first few seconds of streaming that suffice to trigger the RTP-CB. We suggested the use of a pacing buffer in conjunction with the RTP-CB.

Our preliminary results confirm that the effectiveness of the RTP-CB monitoring a video flow is hampered by the bursty behaviour of the coded video. Coupling the RTP-CB with our pacing buffer addresses this problem. Importantly, the RTP-CB can be governed solely by the rules outlined in [5], without imposing ad-hoc RTP-CB rules when a video flow is monitored. Ongoing experiments, considering a wide set of network conditions, will yield further insight.

VII. Acknowledgment

We would like to thank Dr Raffaello Secchi for his valuable help with the experimental setup.

VIII. References

[1] W3C and the IETF, "Real Time Collaboration on the World Wide Web (RTC-WEB)," 2011.
[2] RMCAT IETF working group, "RTP Media Congestion Avoidance Techniques (RMCAT)," *RMCAT Charter*, 21 September 2012.
[3] B. Wang, "Multimedia streaming via TCP: An analytic performance study," ACM Transactions on Multimedia Computing, Communications and Applications, vol. 4, pp. 16, 2008.
[4] M. A. Azad, "A comparative analysis of DCCP variants (CCID2, CCID3), TCP and UDP for MPEG4 video applications," pp. 40-45, ICICT 2009.
[5] C. Perkins and V. Singh, "Multimedia Congestion Control: Circuit Breakers for Unicast RTP Sessions IETF Draft," February 25, 2013.
[6] Y. Ganjali and N. McKeown, "Update on Buffer Sizing in Internet Routers," ACM SIGCOMM Comp. Comm. Review, vol. 36, pp. 67-70, 2006.
[7] J. Watkinson, "The MPEG Handbook: MPEG-1, MPEG-2, MPEG-4," Focal Press, 2004.
[8] A. Keller , "Manual: tc Packet Filtering and netem," ETH Zurich, 2006.
[9] H. Schulzrinne, A. Rao and R. Lanphier , "Real Time Streaming Protocol (RTSP) - RFC 2326," April 1998.
[10] I. Live Networks, "LIVE555 Streaming Media," http://live555.com.
[11] H. Schulzrinne and S. Casner , "RTP profile for audio and video conferences with minimal control- network working group- RFC3551," July2003.
[12] H. Schulzrinne, S. Casner, R. Frederick and V. Jacobson, "RTP: A transport protocol for real-time applications-RFC3550," July 2003.
[13] ESA SUNRISE Project (2000-2001):
http://www.erg.abdn.ac.uk/projects/00esa-sunrise.html