Telecommunication Systems (2022) 79:261–270
https://doi.org/10.1007/s11235-021-00848-0

Application of active queue management for real-time adaptive video streaming

Wladimir Gonçalves de Morais[1] · **Carlos Eduardo Maffini Santos**[1] · **Carlos Marcelo Pedroso**[1]

Accepted: 9 October 2021 / Published online: 24 November 2021

Abstract
Video streaming currently dominates global Internet traffic. Live streaming broadcasts events in real-time, with very different characteristics compared to video-on-demand (VoD), being more sensitive to variations in delay, jitter, and packet loss. The use of adaptive streaming techniques over HTTP is massively deployed on the Internet, adapting the video quality to instantaneous condition of the network. Dynamic Adaptive Streaming over HTTP (DASH) is the most popular adaptive streaming technology. In DASH, the client probes the network quality and adjusts the quality of requested video segment according to the bandwidth fluctuations. Therefore, DASH is an over-the-top application using unmanaged networks to distribute content in the best possible quality. In order to maintain a seamless playback, VoD applications commonly use a large reception buffer. However, in live streaming, the use of large buffers is not allowed because of the induced delay. Active Queue Management (AQM) arises as an alternative to control the congestion in router's queue, pressing the traffic sources to reduce their transmission rate when it detects incipient congestion. In this article, we evaluate the performance of recent AQM strategies for real-time adaptive video streaming. Furthermore, we propose a new AQM algorithm to improve the user-perceived video quality. The results show that the proposed method achieves better performance than competing AQM algorithms and improves the video quality in terms of average peak signal-to-noise ratio while keeping the fairness among concurrent flows.

Keywords Multimedia communication · Live streaming · Active queue management · Video streaming · DASH

1 Introduction

Over the past few decades, the increased demand for video transmission has put pressure on the evolution of the network infrastructure. While the core of the Internet has evolved to accommodate demand, most of the observed congestion occurs in access networks, especially in the last mile [5,13].

Dynamic Adaptive Streaming over HTTP (DASH) has become a de facto standard for video-on-demand (VoD) and is also widely used for live streams [7]. In DASH, videos are divided in segments and encoded with different bitrates/quality using H.264 advanced video coding. The segments are stored on the server, and the client chooses which segment is most appropriate for transfer, depending on the estimated instantaneous network capacity [25].

As an alternative to reduce packet discard, especially in the last mile, where there are network bottlenecks [20,21], Internet service providers (ISPs) are looking to increase the router's buffer length in an attempt to better accommodate traffic. This trend was driven by cheaper memory prices [15]. The excessive buffering of packets may dramatically increase end-to-end latency and jitter, severely impairing the perceived quality of live video transmission. The phenomenon is called bufferbloat [14].

Active queue management (AQM) is a proactive congestion control scheme by which the network sends information to the traffic sources if incipient congestion is detected [1,2]. In response, the traffic sources reduce their transmission rates, which prevents the collapse of the buffers and avoids the network congestion [17,26]. The use of AQM methods also leads to a better fairness in resource distribution. Stud-

✉ Carlos Eduardo Maffini Santos
carlos.maffini@ifpr.edu.br

Wladimir Gonçalves de Morais
wladimirmorais@ufpr.br

Carlos Marcelo Pedroso
pedroso@eletrica.ufpr.br

[1] Department of Electrical Engineering, Federal University of Parana, Curitiba, Brazil

ies have demonstrated that the use of AQM can improve the perceived quality of video streaming application, mainly in congested networks [18].

Active Queue Management algorithms have become important regulators of congestion and provide fairness between flows. TCP (Transport Control Protocol) adapts its transmission rate to match the available network capacity [7]. The first AQM algorithms were originally designed to explore TCP rate adaptation capability to prevent network congestion. Several AQM methods are reported in the literature to address this issue. Despite decades of research, few of them explore how adaptive video traffic interacts with the AQMs [20]. Most AQMs randomly discard packets during congestion periods, regardless of the nature of the traffic pattern. The self-similar behavior of video traffic and the problem of bufferbloat may lead to a large increase in packet delay. This in turn impairs the quality of live streams, as the packets have strict delay limits. Live video streaming traffic has predictable features, which can be used to implement a new class of AQM algorithms. Considering DASH live streaming, AQM could assist lower layers to prevent congestion, performing early discards and forcing the client to request lower-quality segments leading to a smoother adaptation procedure, resulting in better average video quality.

The growing importance of live video streaming, as a result of behavioral changes due to the COVID-19 pandemic, motivates the deployment of new AQM techniques specially designed for DASH applications. There is currently no AQM designed for live DASH applications. In this article we proposed a new AQM for live DASH and compare it with the main available methods. We also evaluate the performance of recent AQM methods for real-time DASH. The proposed method uses the packet queue time to perform a random early discard. This random early discard is designed to induce the DASH quality adaptation to anticipate congestion, therefore improving the average quality as perceived by users and the fairness between video streams. Performance evaluation was done streaming real videos in a simulated network implemented with Network Simulator version 3 (NS-3) [32]. The quality of the received video was estimated with the peak signal-to-noise ratio (PSNR). We present performance comparisons with state-of-the-art AQM algorithms for real-time DASH. Random early detection (RED), adaptive RED (ARED), controlled queue delay (CoDel), proportional integral controller enhanced (PIE), and the proposed method were evaluated. To the best of our knowledge, there are no previous studies on evaluating the impact of AQMs in user-perceived video quality for live dynamic adaptive video streaming. Results show that the average PSNR varies greatly, depending on the AQM method implemented in the routers, and as the network congestion increases. The proposed method outperforms the competing AQMs, especially in situations of high network utilization.

The rest of this article is organized as follows: Sect. 2 gives an overview of MPEG-DASH technology. Section 3 presents the main AQM methods available for use in access networks to support DASH. The proposed method is described in Sects. 4 and 5 presents the performance evaluation. The conclusions are presented in Sect. 6.

2 Adaptive video transmission with DASH

Modern video distribution platforms across the Internet have adopted DASH as the primary video delivery technique [21]. In order to propose a standard for video streaming over HTTP, MPEG (Moving Picture Expert Group) created a solution called MPEG-DASH [36]. MPEG-DASH specifies that the video is encoded in different bitrates/qualities, divided into segments, and stored in a server. The client dynamically looks for the segment that best adapt to network congestion conditions, based on metrics such as throughput, delay, jitter, and playback buffer status. Upon initiating a session, the client sends a request to the server requesting the manifest file known as MPD (Media Presentation Description). MPD contains the information related to available segments, video resolution, bit rates, and timing. The client dynamically looks for the segments that best match current network condition. Figure 1 shows the basic operating scheme of DASH technology.

In VoD systems, the receive buffer must have room to store 20 to 30 seconds of video [7] to ensure a continuous playback experience. However, in real-time video streaming, such long delays are not possible [27]; average packet delay of a few seconds is not suitable for this type of applications. Thus, decreasing the size of the buffer to less than 2 seconds of video requires the player to respond quickly to changes in the network congestion state [7]. This could impair the user-perceived image quality. DASH systems use adaptation mechanisms that mainly involve the application layer, an unexplored feature by the available AQM methods. Therefore, AQM can be used as important congestion regulators in order to cooperate with the estimation and adaptation algorithm of DASH applications.

3 AQM methods

RED [12] was one of the first AQM methods. RED tracks the average queue size through an exponential weighted moving average. The method uses two main thresholds, min_{th} and max_{th}. If the average value of the queue size is below min_{th}, no packet is discarded. If this value is greater than min_{th}, but lower than max_{th}, packets can be discarded with probability given by $p_a = p_b/(1 - count.p_b)$, with $p_b = max_p(avg - min_{th})/(max_{th} - min_{th})$, where max_p

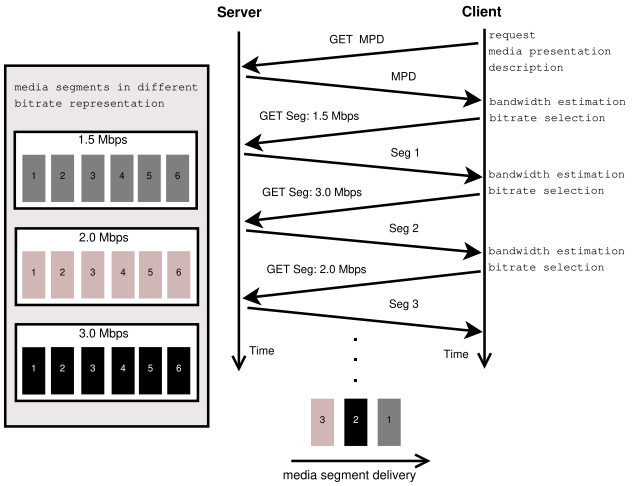

Fig. 1 DASH operation scheme

is the maximum discard probability. If average queue size exceeds max_{th}, all received packets are discarded. The algorithm is presented in Algorithm 1. RED can also be used in conjunction with the ECN extension of the IP protocol enabling notification to traffic sources in case of network congestion.

Algorithm 1: Random Early Detection [12]

For each packet arrival;
calculate the average queue size avg;
if $min_{th} < avg < max_{th}$ **then**
  calculate probability p_a;
  with probability p_a: mark the arriving packet;
end
else if $max_{th} \leq avg$ **then**
  mark the arriving packet;
end

A well-known weakness of RED is that the throughput depends on the traffic load and the RED parameters [11]. RED does not perform well when the average queue size becomes larger than max_{th}, reducing throughput and increasing packet dropping. Adaptive RED [9] is an alternative to improve RED, through dynamic adjustment of max_p according to instantaneous network conditions, improving the robustness of the original algorithm.

In ARED, max_p is adapted using the queue length, improving throughput and reducing packet loss by keeping the average queue length away from max_{th}, as presented by Algorithm 2. Adaptive RED slowly adapts max_p over time scales greater than a typical round-trip time, keeping the average queue length within a target range half-way between min_{th} and max_{th}. As a result, ARED is able to maintain a steady average queue length. An alternative for max_p adaptation was later proposed by Floyd et al. [11]. Authors indicate that the algorithm is not an optimal solution, but seems to work well in a wide range of scenarios.

Algorithm 2: Adaptive RED [9]

Every Q_{avg} update ;
if $min_{th} < Q_{avg} < max_{th}$ **then**
  status $\leftarrow$ between;
end
if $Q_{avg} < min_{th}$ && $status! = Below$ **then**
  status $\leftarrow$ Below;
  $max_p = max_p/\alpha$;
end
if $Q_{avg} > max_{th}$ && $status! = Above$ **then**
  status $\leftarrow$ Above;
  $max_p = max_p * \beta$;
end

Adapting max_p to maintain the average queue size within a target range is one issue of RED addressed by ARED. For high congested links, RED and ARED schemes induce a higher delay, increase the number of discarded packets and are not efficient to keep a good throughput. In order to solve those problems, Patel and Karmeshu [31] suggested a new method to evaluate the discard probability: if the average queue size is between min_{th} and max_{th}, packets are discarded with probability given by $p_2 = 1 - \{p_1[-log(p_1)]/(count + 1)\}$, with $p_1 = p_b$. The results show that the AQM scheme prevents the queue length from exceeding max_{th}, increasing the throughput. Also, the scheme maintains the average queue length in lower levels because a better selection of packet discard probability, decreasing end-to-end delay in situations of network congestion.

It is know that RED parameters need to be tuned to work well in a diversify of scenarios. The work presented by Shalabh et al. [6] introduces an optimization technique based on stochastic approximation to tune RED's parameters in order to achieve high throughput and low loss-rate. The results presented by the authors show that the AQM achieves better throughput and lower loss-rate than RED, ARED, MRED and, TRED in several situations of network congestion.

Bufferbloat is the undesirable latency caused by the excessively large and frequently full buffers in network routers. Large buffers have been inserted all over the Internet without sufficient thought or testing [14]. This phenomenon causes high latency and jitter, with negative effects on the applications. CoDel (Controlled Queue Delay) [28] is an AQM designed to provide a solution for the bufferbloat problem. Its operation is based on the control of the delay in the queue by creating a timestamp of packet arrival time. CoDel uses two key variables: *target* and *interval*. In conformity with RFC 8289 [29], ideal values of *target* is 5–10% of the con-

nection RTT (Round Trip Time). As most unbloated RTTs in open terrestrial-based Internet have a ceiling of 100 ms [8], default values of *interval* and *target* are set as 100 and 5 ms, respectively. Each *interval* CoDel computes the delay of all packets dequeued for forwarding. If minimum queue delay is lower than *target*, or the buffer contains fewer than MTU worth of bytes, packets are not dropped. If minimum delay is greater than *target*, CoDel enters in drop mode, and a single packet is discarded. Then the next *interval* is set in accordance with the inverse square root of the number of successive intervals in which CoDel is in drop mode. Thus, default sequence of the *interval* in drop mode is given by $100, 100/\sqrt{2}, 100/\sqrt{3}, \ldots$. Once the minimum delay of all packets in *interval* goes below *target*, CoDel exits the drop mode, no packets are discarded, and *interval* returns to its default value.

PIE [30] is a method that combines the benefits of RED and CoDel. PIE is a lightweight-design controller with the aim to control the average queueing latency to a reference value. The design does not require per-packet extra processing and is simple to implement. Like CoDel, the parameters are self-tuning. PIE randomly drops a packet at the onset of the congestion similar to RED; however, congestion detection is based on the queueing latency like CoDel instead of the queue length like conventional AQM schemes. PIE discards packets randomly according to a probability. The drop probability is computed using the current estimation of the queueing delay and the delay trend, that is, whether the delay is getting longer or shorter. The PIE algorithm updates the drop probability periodically using Little's law (queue delay is given by the ratio between queue size and arrival rate) and the delay threshold. In addition, the scheme uses a maximum allowed value for packet bursts to be allocated in the buffer. Auto-tuning of parameters is used not only to maintain stability but also to respond fast to sudden changes. Pan et al. [30] argue that PIE design is stable for an arbitrary number of flows with heterogeneous RTTs and achieves low latency and high link utilization under various congestion situations.

Emerging AQM schemes such as PIE and CoDel are being progressively deployed either at the ISP-end or home gateway to prevent bufferbloat. Kua and Armitage [19] propose the joint use of AQM strategies (as PIE or CoDel) and intra-chunk parallel connections to improve the user-perceived quality of DASH. Their method, uses N concurrent TCP connections to retrieve different parts of video segments. Thus, DASH connections achieve better share of bandwidth in the presence of competing traffic. The results show that the use of a fair-queue strategy with CoDel (FQ-CoDel) enables DASH chunklets to attain the best throughput multiplication effect, resulting in a better user experience in the presence of competing elastic flows. Besides, the results show that chunklets are advantageous when competing with multiple flows. However, the number of chunklets need to be increased as the number of competing flows increase. The main problem of chunklets are the consumption of server resources and the unnecessary starvation of concurrent flows. Kua et al. [22] propose the use of adaptive chunklets to dynamically adjust the method to achieve the best possible user experience without starve other flows unnecessarily. In other paper, Kua et al. [23] proposed a client-side application for detecting the bottleneck AQM scheme. Through an intra-chunk adaptive rate measurements, the application is able to distinguish AQM scheme from conventional FIFO and FlowQueue-like AQMs. And finally, Kua et al. [20] experimentally characterize and evaluate the impact of bottlenecks using PIE and FQ-CoDel AQM schemes on DASH streams. The results show that PIE's higher burst tolerance provides better streaming quality for single DASH stream over moderate to high RTT paths.

Abbas et al. [1] presents an AQM scheme to improve fairness between flows, identifying and penalizing unresponsive flows, since they keep on sending packets despite the congestion indications. Called CHOKeH, the algorithm reduces the drop rate of responsive flows without the need to maintain any per-flow state. The basic idea of CHOKeH is similar to RED, using the average queue size to measure the network congestion and two thresholds, min_{th} and max_{th}. For each packet arrival, if the average queue size is between min_{th} and max_{th}, CHOKeH splits the current queue size in two regions of equal length, the rear and front regions. CHOKeH randomly choose the drop-candidates of each region with differently probabilities. This procedure ensures that high bandwidth unresponsive flows with many recent arrivals are penalized. The results show that the CHOKeH achieves better throughput and a stable behavior of average queue size than competing AQMs.

4 Proposed method

We propose a random packet discard strategy based on the expected packet queue time. The expected time in queue for the i-th packet is given by $t_i = \sum_{j=1}^{i} b_j/C$, where b is the packet size, and C is the link rate; t_i is evaluated for every packet received. If t_i is lower than T_{min}, the packet is enqueued. If t_i exceeds T_{min}, but is smaller than the upper threshold T_{max}, the algorithm performs a random drop. The drop probability is given by $p = t_i/T_{\max}$. If t_i exceeds T_{max}, the packet is discarded. The queuing policy is first-in, first-out (FIFO). The proposed algorithm is presented in Algorithm 3.

In response to an early packet discard, the DASH client reduces the quality of the next segment to be requested. RED uses the same principle to induce the decrease of TCP congestion window. However, RED cannot be applied directly because the thresholds are setted considering the queue size

Algorithm 3: Proposed algorithm

```
receivePacket(&P_i);
t_i ← Σ_{j=1}^{i} b_j / C;
if t_i < T_min then
    enqueue(P_i);
end
if T_min ≤ t_i ≤ T_max then
    p ← t_i / T_max;
    if random() < p then
        drop(P_i);
    else
        enqueue(P_i);
    end
end
if t_i > T_max then
    drop(P_i);
end
```

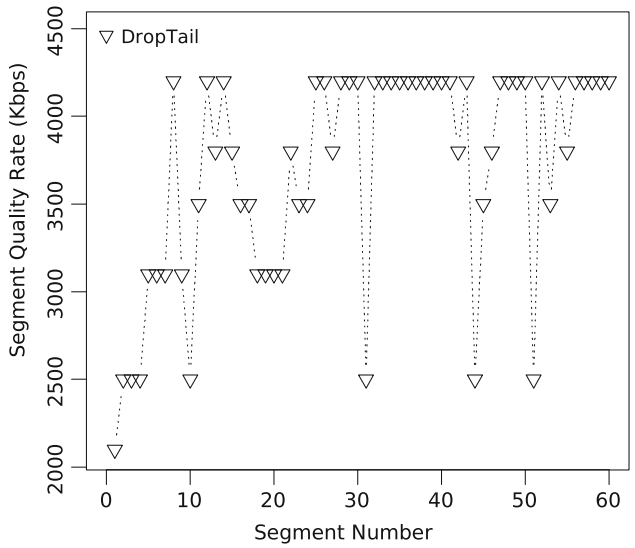

Fig. 2 Transition between quality of segments along of a typical DASH transmission

rather than the packet queue time. We expect to improve the average video quality perceived by users, as well as a better justice among concurrent DASH flows. Figure 2 illustrates the segment quality in a DASH flow through a congested link (85% of occupation) using FIFO and an infinite queue size. In Fig. 2, the reader can note the sudden quality transitions.

This is due to the deadline to reproduce real-time video and the competition of multiple video streams. With the proposed method we intend to smooth the transitions between segment qualities. As result, we expect an improvement of average PSNR and segment quality transitions less noticeable to the user.

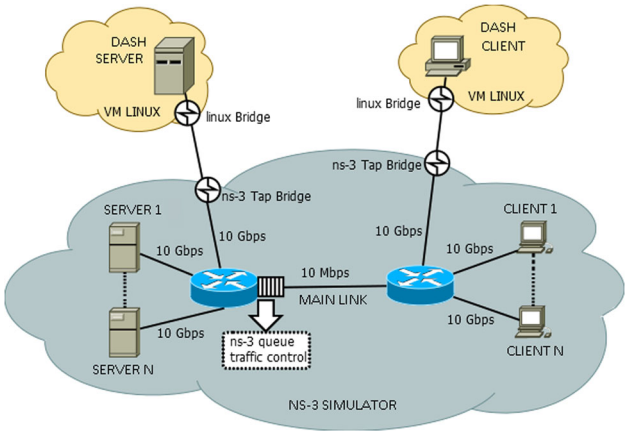

Fig. 3 Scenario used in simulations

5 Performance evaluation

The performance evaluation was done by integrating real DASH server and client into a NS-3 simulation. DASH server and client were implemented using virtualization. The proposed method was implemented in the NS-3. Network Simulator version 3 is an open-source network simulator, available primarily for research, enabling the development of realistic network models.

The simulation uses a dumbbell topology with two main routers, connected by a 10 Mbps link. The remaining links were set to 1 Gbps. Thus, the connection between the two routers simulates a bottleneck in the access network.

The DASH server and client were implemented using the GPAC Multimedia Open Source Project [24], installed in virtual machines and attached to the simulated scenario. This enabled us to assesses the impact of network congestion in real live video streaming. Figure 3 presents the scenario implemented for the simulations.

We use six full-HD (high definition, 1920 × 1080) raw video sequences in the tests: Big Buck Bunny (BB), Sunflower (SF), Rush Hour (RH), Pedestrian Area (PA), and Riverbed (RB), all publicly available [34].

Table 1 summarizes the characteristics of the videos sequences used, such as total length, number of frames, genre, temporal perceptual information (TI), spatial perceptual information (SI), and format of raw video source. TI indicates the amount of temporal chances of a video sequence, with higher values for more spatially complex scenes [37]. SI indicates the amount of spatial detail of a picture, with higher values for high motion scenes [37]. The video sequences were selected to cover a range of different genres, TI, and SI, improving diversity in the performance evaluation.

BB is the longest video sequence used in the tests, with 1440 frames, displaying the highest TI and SI. SF sequence uses a fixed camera to capture a bee in the foreground and a

Table 1 Characteristics of videos used in performance evaluation

Video	Length (s)	Frames	Genre	TI	SI	Raw format
BB	60	1440	Animation	67.02	73.86	YUV420
SF	20	500	Nature	26.19	39.38	YUV420
RH	20	500	Scene	18.86	26.72	YUV420
PA	15	375	Scene	21.08	37.08	YUV420
RB	10	250	Nature	29.66	39.45	YUV420
TP	19	570	Sports	27.13	57.94	YUV422

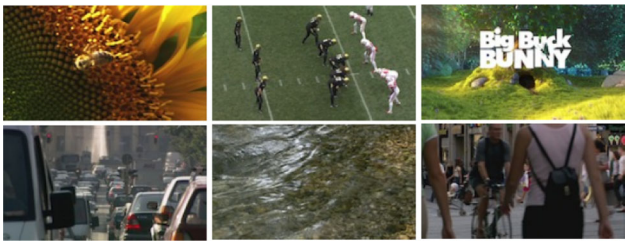

Fig. 4 Intermediate frame of SF, TP, BB, PA, RB, and RH videos [34]

Table 2 Maximum PSNR for videos

Video	Average PSNR
BB	37.0
SF	38.8
RH	40.4
TP	40.3
PA	39.6
RB	30.7

flower in the background. RB sequence uses a fixed camera to capture water movement, showing the third highest SI. RH sequence shows vehicle traffic and heat waves during rush hour in the city of Munich. PA shows people passing by very close to the camera. RH and PA present the lowest TI and SI among the videos. TP illustrates fast-moving players on a soccer field and presents the second highest SI.

Figure 4 illustrates an intermediate frame of each video used. The encoding of the videos was done offline using the FFmpeg [10] tool. Live segment generation was performed following the profile Live-H.264 according to the MPEG-DASH standard [35]. The video server provides the manifest file and several representations of video segments using an Apache web server. Live video segments were generated with length of 1 second [25]. The Group of Picture (GOP) was set to six frames, with two B-frames between I- and P-frames. Because each video was encoded at a rate of 24 fps, each single 1-second DASH segment contains four GOPs [35]. Segments were encoded using the following representations: 2.1, 2.5, 3.1, 3.5, 3.8 and 4.2 Mbps, compatible with other studies [21,25], with HD resolution.

Background traffic sources generate packets of 1500 bytes using TCP, according to the Poisson Pareto Burst Process (PPBP) model [4]. PPBP is based on the overlapping of multiple bursts whose length follows a Pareto distribution. Pareto probability density function is given by $P(X = x) = (\alpha\beta^{\alpha})/x^{(\alpha+1)}$. PPBP can be used to simulate video traffic with self-similar characteristics with Hurst parameter given by $H = (3-\alpha)/2$ for $1 < \alpha < 2$ [3]. The parameters α and β were set to produce self-similar background traffic with Hurst parameter of 0.7.

Several simulations were performed by varying (i) the AQM method used in queue and (ii) number of background traffic sources. For each video and traffic intensity, in addition to the proposed method and Droptail, the following AQMs were tested: RED, ARED, CoDel, and PIE. The buffer size was set to maximum capacity of 500 packets for RED, ARED, CoDel, and PIE. Droptail uses an infinite-capacity buffer, which could lead to the bufferbloat problem.

The received video quality was estimated using the average PSNR. PSNR is a metric that assesses the similarity between two images, computing the mean square error (MSE) of each pixel between the original and the received images. The PSNR is evaluated frame by frame using the MSE given by

$$\text{MSE} = \frac{1}{rc}\sum_{i=1}^{r}\sum_{j=1}^{c}[X_o(i,j) - X_r(i,j)]^2 \tag{1}$$

where r and c represent, respectively, the number of rows and columns of the image, and $X_o(i,j)$ and $X_r(i,j)$ represent the luminance of pixel (i,j) of the original and received frames, respectively. The PSNR can be obtained using

$$\text{PSNR} = 20\log_{10}\left(\frac{\text{MAX}_I}{\sqrt{\text{MSE}}}\right) \tag{2}$$

where MAX_I represents the maximum value of pixel intensity. For the videos in consideration, $\text{MAX}_I = 255$. Images with more similarity result in higher PSNR values [33]. If PSNR is greater than 37 dB, the perceived video quality is excellent, and if PSNR is lower than 20 dB, the quality is

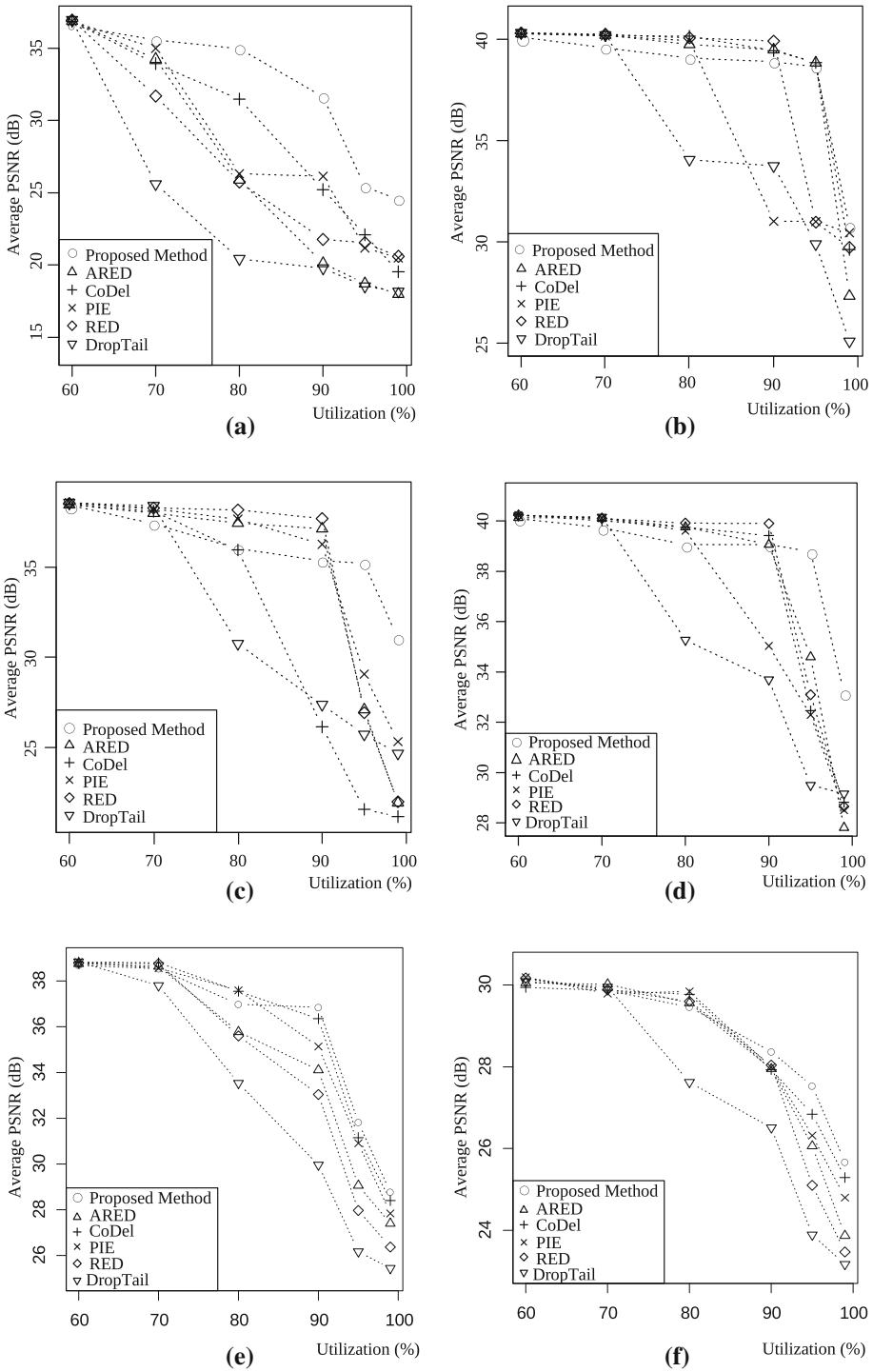

Fig. 5 Average PSNR for video **a** BB, **b** RH, **c** SF, **d** TP, **e** PA, and **f** RB

very poor [38]. As a reference, Table 2 presents maximum possible PSNR for all videos after encoding using MPEG4.

Figure 5 presents the simulation results, with average PSNR (dB) in vertical axis and bottleneck link utilization in horizontal axis. Figure 5a indicates that the proposed method greatly outperformed the competing AQMs. CoDel presented the second best performance, but with average PSNR of 4 to 7 dB lower than the proposed method. Figure 5b–d presents the simulation results for RH, SF, and TP, respectively. It is possible to see that the proposed method presents a slightly worse PSNR for bottleneck utilization of 95%, but better average PSNR above this level. Figure 5e, f shows the simulation results for PA and RB. It is possible to notice that the proposed method presents better average PSNR, mainly for high link utilization. In general, the CoDel and PIE algorithms performed better than RED and ARED, but the proposed method

Table 3 Jain justice index for average throughput

	60%	70%	80%	90%	95%	99%	Average
Droptail	0.900	0.882	0.906	0.933	0.804	0.802	0.871
RED	0.965	0.949	0.945	0.948	0.956	0.925	0.948
ARED	0.947	0.955	0.974	0.964	0.954	0.943	0.956
CoDel	0.988	0.931	0.907	0.923	0.960	0.968	0.946
PIE	0.965	0.948	0.948	0.957	0.920	0.927	0.944
Proposed method	0.951	0.961	0.963	0.960	0.967	0.939	0.957

Table 4 Jain justice index for average delay

	60%	70%	80%	90%	95%	99%	Average
Droptail	0.771	0.903	0.653	0.616	0.680	0.550	0.695
RED	0.887	0.691	0.674	0.837	0.597	0.853	0.757
ARED	0.945	0.966	0.720	0.768	0.699	0.895	0.832
CoDel	0.918	0.956	0.891	0.958	0.973	0.936	0.939
PIE	0.941	0.886	0.836	0.921	0.962	0.908	0.909
Proposed method	0.974	0.902	0.911	0.886	0.942	0.974	0.932

achieved better results. After 70% of link utilization, Droptail shows the worst performance for all videos. This is due the bufferbloat phenomenon that increases packet latency, which could cause segment expiration and often freezes video playback.

We also evaluate the fairness of delay and throughput between competing video streams. The fairness was estimated using Jain's fairness index [16], given by

$$J(x_1, x_2, x_3, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} x_i^2} \tag{3}$$

where x is the metric under study, and n is the sample size. $J(x_1, x_2, x_3, \ldots, x_n)$ is a real number between 0 and 1, where 1 indicates the best possible fairness level.

Tables 3 and 4 present the results of Jain's justice for average throughput and average delay between competing streams for BB. The utilization of bottleneck link varied from 60 to 99% by increasing the background traffic sources. Considering the throughput, the proposed method achieved a better fairness, followed by ARED, RED, CoDel, PIE, and Droptail. In this case, CoDel and PIE were outperformed by RED and ARED. Considering the delay, CoDel presents a better fairness, followed by the proposed method, PIE, ARED, RED, and Droptail. Codel and PIE were designed to limit the queue delay, and this result could be expected. The proposed method achieved a justice index very close to CoDel. These results could also explain the better average PSNR achieved by the proposed method and the good performance of Codel and PIE in real-time video streaming. Droptail achieves the worst results in all cases. Results also indicate that the use of RED for real-time video streaming, although better than Droptail, should be avoided.

6 Conclusions

The current generation of DASH client player requires large buffers in order to store a significant number of segments to avoid video freezing. Considering real-time video streams, the use of large buffers is not allowed, as large buffers also mean higher delay. The choice of the segment to be received influences the quality as perceived by the user. Choosing video segments with better quality increases network congestion, which paradoxically could worsen the user-perceived quality due the increase of delay and unfair sharing of resources.

In this article, we study the performance of several AQM algorithms to support real-time video streaming. The performance evaluation was done by combining computer simulation and real video streaming. We also proposed a new AQM for real-time video streaming using the packet expected queue time as a criterion for early discard.

Results indicate that the proposed method achieved a better average PSNR for real-time video streaming, followed by CoDel and PIE. The use of the proposed method allows the client to adapt before congestion, lowering quality in advance of network congestion, resulting in higher average PSNR, and good fairness of delay and throughput between competing flows. Based on results, the use of RED or ARED is not recommended for real-time video streaming, and the use of Droptail should also be avoided.

References

1. Abbas, G., Manzoor, S., & Hussain, M. (2018). A stateless fairness-driven active queue management scheme for efficient and fair

bandwidth allocation in congested internet routers. *Telecommunication Systems, 67*(1), 3–20.
2. Adams, R. (2013). Active queue management: A survey. *IEEE Communications Surveys Tutorials, 15*(3), 1425–1476.
3. Addie, R. G., Neame, T. D., & Zukerman, M. (2002). Performance evaluation of a queue fed by a Poisson pareto burst process. *Computer Networks, 40*(3), 377–397.
4. Ammar, D., Begin, T., & Guerin-Lassous, I. (2011). A new tool for generating realistic Internet traffic in NS-3. In *Proceedings of the 4th international ICST conference on simulation tools and techniques* (pp. 81–83). Brussels, Belgium.
5. Bajpai, V., Eravuchira, S. J., & Schönwälder, J. (2017). Dissecting last-mile latency characteristics. *SIGCOMM Computer Communication Review, 47*(5), 25–34.
6. Bhatnagar, S., & Patel, S. (2018). Karmeshu: A stochastic approximation approach to active queue management. *Telecommun Systems, 58*(1), 89–104.
7. Bouten, N., Claeys, M., Latré, S., Famaey, J., Leekwijck, W. V., & Turck, F. D. (2014). Deadline-based approach for improving delivery of SVC-based HTTP adaptive streaming content. In *2014 IEEE network operations and management symposium (NOMS)* (pp. 1–7).
8. Dischinger, M., Haeberlen, A., Gummadi, K. P., & Saroiu, S. (2007). Characterizing residential broadband networks. In *Proceedings of the 7th ACM SIGCOMM conference on internet measurement, IMC '07* (pp. 43–56). ACM, New York, NY, USA.
9. Feng, W., Kandlur, D. D., Saha, D., & Shin, K. G. (1999). A self-configuring RED gateway. In *IEEE INFOCOM '99 conference on computer communications. proceedings of eighteenth annual joint conference of the IEEE computer and communications societies* (Vol. 3, pp. 1320–1328).
10. FFmpeg Developers: FFmpeg multimedia framework (2019). https://www.ffmpeg.org.
11. Floyd, S., Gummadi, R., & Shenker, S. (2001). Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Technical report. AT&T Center for Internet Research at ICSI.
12. Floyd, S., & Jacobson, V. (1993). Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking, 1,* 397–413.
13. Genin, D., & Splett, J. (2013). Where in the internet is congestion? Computing Research Repository (CoRR). arXiv:1307.3696.
14. Gettys, J., & Nichols, K. (2011). Bufferbloat: Dark buffers in the internet. *ACMqueue, 9*(11), 40–54.
15. Grigorescu, E., Kulatunga, C., & Fairhurst, G. (2013). Evaluation of the impact of packet drops due to AQM over capacity limited paths. In *2013 21st IEEE international conference on network protocols (ICNP)* (pp. 1–6).
16. Jain, R. K. (1991). *The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling*, 1st edn (Vol. 1). Wiley.
17. Kahe, G., & Jahangir, A. H. (2019). A self-tuning controller for queuing delay regulation in TCP/AQM networks. *Telecommunication Systems, 71*(2), 215–229.
18. Kim, H. J., Park, P. K., Yoon, H. S., & Choi, S. G. (2011). QOS-aware active queue management scheme for multimedia services. In *13th International conference on advanced communication technology (ICACT2011)* (pp. 1037–1042).
19. Kua, J., & Armitage, G. (2017). Optimising DASH over AQM-enabled gateways using intra-chunk parallel retrieval (chunklets). In *2017 26th International conference on computer communication and networks (ICCCN)* (pp. 1–9).
20. Kua, J., Armitage, G., & Branch, P. (2016). The impact of active queue management on dash-based content delivery. In *2016 IEEE 41st conference on local computer networks (LCN)* (pp. 121–128).
21. Kua, J., Armitage, G., & Branch, P. (2017). A survey of rate adaptation techniques for dynamic adaptive streaming over HTTP. *IEEE Communications Surveys Tutorials, 19*(3), 1842–1866.
22. Kua, J., Armitage, G., Branch, P., & But, J. (2019). Adaptive Chunklets and AQM for higher-performance content streaming. *ACM Transactions on Multimedia Computing, Communications, and Applications, 15*(4), 1–24.
23. Kua, J., Branch, P., & Armitage, G. (2020). Detecting bottleneck use of pie or fq-codel active queue management during dash-like content streaming. In *2020 IEEE 45th conference on local computer networks (LCN)* (pp. 445–448). IEEE Computer Society.
24. Le Feuvre, J., Concolato, C., & Moissinac, J. C. (2007). GPAC: Open source multimedia framework. https://gpac.wp.imt.fr.
25. Lederer, S., Muller, C., & Timmerer, C. (2012). Dynamic adaptive streaming over HTTP dataset. In *Proceedings of the 3rd multimedia systems conference* (pp. 89–94). ACM, New York, NY, USA.
26. Li, Y., Gong, X., Wang, W., Que, X., & Ma, J. (2010). An autonomic active queue management mechanism to improve multimedia flow delivery quality. In *2010 International conference on communications and mobile computing* (Vol. 1, pp. 493–497).
27. Lohmar, T., Einarsson, T., Fröjdh, P., Gabin, F., & Kampmann, M. (2011). Dynamic adaptive HTTP streaming of live content. In *2011 IEEE international symposium on a world of wireless, mobile and multimedia networks* (pp. 1–8).
28. Nichols, K., & Jacobson, V. (2012). Controlling queue delay. *ACMqueue, 10*(5), 20–34.
29. Nichols K., J. V. Controlled delay active queue management (2018). Request for Comment 8289 (Proposed Standard).
30. Pan, R., Natarajan, P., Baker, F., & White, G. (2017). Proportional integral controller enhanced (PIE): A lightweight control scheme to address the Bufferbloat problem (2017). Request for Comment 8033.
31. Patel, S. (2019). Karmeshu: A new modified dropping function for congested aqm networks. *Wireless Personal Communications, 104*(1), 37–55.
32. Riley, G. F., & Henderson, T. R. (2010). The NS-3 network simulator. In: K. Wehrle, M. Günes, & J. Gross (Eds.), *Modeling and tools for network simulation* (pp. 15–34). Springer.
33. Santos, C. E. M., Ribeiro, E. P., & Pedroso, C. M. (2014). The application of neural networks to improve the quality of experience of video transmission over ip networks. *Engineering Applications of Artificial Intelligence, 27,* 137–147.
34. Seeling, P., & Reisslein, M. (2012). Video transport evaluation with H.264 video traces. *IEEE Communications Surveys Tutorials, 14*(4), 1142–1165.
35. Sodagar, I. (2011). The MPEG-DASH standard for multimedia streaming over the internet. *IEEE MultiMedia, 18*(4), 62–67.
36. Stockhammer, T. (2011). Dynamic adaptive streaming over HTTP-standards and design principles. In *Proceedings of the second annual ACM conference on multimedia systems* (pp. 133–144). ACM, New York, NY, USA.
37. Telecommunication, I. T. U. Subjective video quality assessment methods for multimedia applications (2008). ITU-T P.910.
38. Torres, A. J. F., Ribeiro, E. P., & Pedroso, C. M. (2016). Predictive delay-centric handover for video streaming over SCTP. *Computer Communication, 87,* 49–59.

Wladimir Gonçalves de Morais received the B.S. degree in Computer Science from Pontifical Catholic University of Goiás, postgraduate in Computer Networks from Federal University of Goiás and the M.S. degree in Electrical Engineering from Federal University of Paraná, from Brazil. He is currently an IT Analyst in the Federal Government in Paraná, Brazil. His research interests include computer networks and video streaming.

Carlos Eduardo Maffini Santos received the B.Eng. degree in electrical engineering from the Pontifical Catholic University of Parana (PUCPR), in 2009, M.Sc. degree in Electrical Engineering from the Federal University of Parana (UFPR), in 2012, and is attending a doctor degree at UFPR. He served as professor at the technical courses at PUCPR, from 2009 to 2015. He is currently a Professor of techincal course in Electronics for high school students at the Federal Institute of Paraná (IFPR). His research interests include data communication, multimedia systems, the Internet technologies, and electronics systems.

Carlos Marcelo Pedroso received the B.Eng. degree in computer engineering from the Pontifical Catholic University of Parana (PUCPR), in 1994, and the D.Sc. degree in Electrical Engineering from the Federal Technological University of Parana (UTFPR), in 2006. He served as the Director of the Computer Engineering Faculty, Pontifical Catholic University of Parana, from 2006 to 2012. He is currently an Associate Professor with the Electrical Engineering Department, Federal University of Paraná (UFPR). His research interests include data communication, modeling and performance evaluation, multimedia systems, the Internet technologies, and quality in engineering education.