Explicit AQM-based Congestion Notification for Improved Adaptive Video Streaming

Frank Loh†, Alan Weber†, Andrea Pimpinella*, Fabio Martignon*, Tobias Hoßfeld†
†Institute of Computer Science, University of Würzburg, Germany; *University of Bergamo, Italy
contact: frank.loh@uni-wuerzburg.de, andrea.pimpinella@unibg.it

***Abstract*—Video streaming accounts for a substantial share of Internet traffic and increasingly competes with background cross-traffic, challenging resilient and resource-efficient service delivery under congestion. Active Queue Management (AQM) controls queue occupancy by marking or dropping packets before buffer overflow. We examine how different AQM schemes, parameter configurations, and network conditions affect playback quality and network performance and introduce an AQM-based Explicit Buffer Notification (EBN) mechanism that exposes congestion information to clients' adaptive bitrate logic. In our simulations, EBN reduces quality switches by approximately 49 %, stalling events by up to 70 %, and streaming-flow packet drops by approximately 99 % compared with the evaluated conventional approaches, while maintaining comparable video quality. These results indicate that explicit AQM feedback can improve playback resilience and transmission efficiency under congestion.**

***Index Terms*—Video Streaming, AQM, Streaming Quality, Quality of Service, Simulation**

I. Introduction

Consumer video streaming accounts for over 70 % of Internet traffic [1] and competes with web browsing and bulk downloads for network resources. Bursty HTTP Adaptive Streaming (HAS) requests and demand peaks during major events can rapidly saturate bottleneck queues, even when average load remains within link capacity [1]. The resulting delay and packet loss can deplete playback buffers, cause quality oscillations and stalls, and reduce resource efficiency through retransmissions or delivery of unused segment data. Larger bottleneck buffers can absorb short traffic bursts but may cause bufferbloat, i.e., persistent queues and excessive delay [2]. Active Queue Management (AQM) addresses this trade-off by marking or dropping packets before overflow. Earlier schemes, such as Random Early Detection (RED) [3] and Adaptive RED (ARED) [4], primarily rely on queue occupancy, whereas PIE [5] and CoDel [6] target queueing delay and can be combined with Flow Queuing (FQ) to improve fairness. These mechanisms provide a basis for balancing throughput and latency with playback resilience and transmission efficiency.

From an application perspective, congestion can deplete the client's playback buffer, causing quality reductions, oscillations, or playback interruptions. These effects reduce service continuity and may waste network resources through packet loss, retransmissions, or unused segment data. Adaptive Bitrate (ABR) algorithms address network variability by scheduling HTTP requests and adapting the requested bitrate to available resources. Joint approaches in which the network directly informs ABR decisions may therefore improve playback resilience, quality, and transmission efficiency.

This work evaluates AQM performance under HAS traffic and makes two contributions. First, we implement, simulate, and compare several AQM schemes with the baseline DropTail discipline. An extensive parameter study across different network conditions identifies configurations that provide favorable trade-offs between network performance and playback resilience. Second, we propose an AQM-based Explicit Buffer Notification (EBN) mechanism that communicates bottleneck queue conditions to the application layer of streaming clients. By integrating network-side congestion information into the ABR algorithm, EBN reduces reliance on packet loss as a congestion indicator. We evaluate EBN with different underlying AQM schemes and compare it with ABR baselines that do not use explicit network-side feedback. Our discrete-event Python simulator, configurations, and evaluation scripts are publicly available[1] to support reproducibility and follow-up work. We provide simulation-based evidence that exposing AQM information through Explicit Congestion Notification (ECN)-style feedback can stabilize ABR and reduce packet-loss-related transmission overhead. Future work will consider additional transports, including CUBIC, BBR, and QUIC, as well as signaling compatible with Low Latency, Low Loss, and Scalable Throughput (L4S). To this end, we address the following two research questions:

RQ1: Which AQM mechanisms and parameter configurations provide favorable trade-offs between network QoS, transmission efficiency, and playback resilience for on-demand video streaming?

RQ2: Can direct queue-congestion notification improve playback resilience and transmission efficiency, and how can it be integrated into ABR?

In the remainder, Section II reviews related work, Section III details the methodology, Section IV provides numerical results, and Section V concludes.

II. Related Work

Numerous studies evaluate AQM and propose optimization strategies tailored to video streaming. In [7], the authors assess the impact of PIE, FQ-CoDel, and FQ-PIE on HAS, showing that while FQ variants reduce RTT and protect ACK flows, they may limit streaming throughput compared to PIE. When

[1]The code is available at: https://github.com/lsinfo3/aqmsim_ecn

Algorithm 1 AQM-based EBN decision logic.

Require: B, C, $\theta_{\min}$, MARK$(\cdot)$, incoming packet p
Ensure: Send explicit buffer notification to streaming client
1: **if** p belongs to a streaming flow **then**
2: **if** MARK(B, p) **then**
3: Send EBN(BUFFER_DEPLETING)
4: **else if** $B < \theta_{\min} \cdot C$ **then**
5: Send EBN(BUFFER_AVAILABLE)
6: **else**
7: Send EBN(BUFFER_STABLE)
8: **end if**
9: **else**
10: **return**
11: **end if**

QUIC is adopted, ARED and RED outperform DROPTAIL in terms of Video Multimethod Assessment Fusion scores and rebuffering duration, while CoDel can degrade QoE under low congestion [8]. Recent work increasingly targets video-centric metrics. MORAIS [9] adapts drop behavior based on packet size and link rate, enabling faster DASH reactions to congestion, while [10] proposes an LSTM-based AQM that predicts queue delay to adjust drop probabilities and improves performance under load. Beyond AQMs, network-assisted and application-aware approaches have also emerged. These include client-side AQM detection [11], as well as reinforcement learning for flow prioritization [12].A recent example is Network Delivery Time Control (NDTC) [13], which targets low-latency interactive video flows (e.g., cloud gaming) and adapts the sending rate based on delivery-time feedback. While NDTC focuses on real-time media, we consider on-demand HAS and ask how access-router AQM signals can be exposed to the player to reduce stalls and oscillations under sustained congestion. The IETF has also standardized the Low Latency, Low Loss, Scalable Throughput (L4S) architecture [14], together with DualQ coupled AQM [15]. L4S relies on higher-frequency ECN marking to keep latency low while maintaining throughput. Because EBN is driven by ECN marks rather than loss, it is conceptually aligned with L4S-style feedback. We discuss in Section V how a continuous, marking-density EBN variant could better exploit this regime.

III. METHODOLOGY

We first describe how we adapt traditional AQM logic with EBN, and then detail our simulation and evaluation approach.

A. AQM-Integrated Explicit Buffer Notification

Traditional AQM schemes use packet loss or ECN marking and rely on end-to-end TCP congestion control to reduce the sending rate. In contrast, our EBN approach uses the AQM state to explicitly inform the streaming client at the application layer. The congestion indication can be conveyed through ECN Congestion Experienced (CE) marks, while the remaining EBN states can use lightweight client-visible signaling. Nevertheless, packet drops may still occur under overload. The proposed approach operates on a per-packet basis, as shown in Algorithm 1. The router first filters incoming traffic so

Algorithm 2 Client-side quality adaptation based on EBN.

Require: q_{prev}, $\mathcal{Q}$, e, t, t_{last}, Δt, $Hybrid$
Ensure: Next quality level q_{next}
1: $q_{\min} \leftarrow \min(\mathcal{Q})$
2: $q_{\max} \leftarrow \max(\mathcal{Q})$
3: **if** $e \neq \emptyset$ **and** $t \geq t_{\text{last}} + \Delta t$ **then**
4: **if** $e =$ BUFFER_AVAILABLE **then**
5: $q_{\text{next}} \leftarrow \min(2 \cdot q_{\text{prev}}, q_{\max})$
6: **else if** $e =$ BUFFER_DEPLETING **then**
7: $q_{\text{next}} \leftarrow \max(\frac{q_{\text{prev}}}{2}, q_{\min})$
8: **else if** $e =$ BUFFER_STABLE **then**
9: **if** $Hybrid$ is $true$ **then**
10: $q_{\text{next}} \leftarrow$ ABR logic from [17]
11: **else**
12: $q_{\text{next}} \leftarrow q_{\text{prev}}$
13: **end if**
14: **end if**
15: Clear stored EBN
16: $t_{\text{last}} \leftarrow t$
17: **return** q_{next}
18: **end if**
19: **return** q_{prev}

that EBN signals are generated only for packets p belonging to an active video-streaming flow, for example using flow separation [16] (line 1). When the underlying AQM logic, represented by the dynamic marking function MARK(B, p), determines that a video packet should be marked, the router conveys a BUFFER_DEPLETING signal to the client (line 3), for example through an ECN CE mark. This ensures that the ABR client reacts only when the AQM logic marks a packet belonging to an active streaming flow, avoiding signaling for unrelated background traffic. If a marked packet does not belong to a streaming flow, the algorithm returns without generating an EBN signal (lines 9–10). To support quality increases under favorable conditions, the algorithm also monitors the buffer occupancy B relative to its capacity C. If no video packet is marked and B is below the threshold $\theta_{\min}C$, it generates a BUFFER_AVAILABLE signal (line 5), indicating that the client may increase the requested video quality. If neither congestion nor underutilization is detected, a BUFFER_STABLE signal is sent (line 7), maintaining regular feedback and reducing the client's reliance on indirect network-state estimates.

B. EBN-Driven Adaptive Bitrate Logic

Algorithm 2 describes the client-side adaptation mechanism that processes incoming notifications and adjusts the video quality level q. Its objective is to maintain high visual quality while limiting playback stalls through timely responses to network feedback. A cooldown interval Δt prevents frequent quality changes: adaptation occurs only when a valid notification e is available and sufficient time has elapsed since the previous change, i.e., $t \geq t_{\text{last}} + \Delta t$ (line 3). When these conditions are met, the client applies a multiplicative increase/decrease strategy (lines 4–14). A BUFFER_AVAILABLE notification doubles the previous quality level, subject to the upper bound $q_{\max}$ (lines 4–5). Conversely, BUFFER_DEPLETING halves the previous quality level to reduce network load, subject to the lower bound $q_{\min}$ (lines 6–7). For BUFFER_STABLE, we evaluate two alternatives (lines 8–13): i) maintaining the current quality level to prioritize stability, and ii) a *Hybrid*

Table I: Simulation parameters and corresponding ranges.

Parameter	Values	Description
Simulation duration	600 s	Length of a simulation run
Router delay	20 ms	Bottleneck link delay
Target buffer size	20 s	Client playback buffer length
Base bottleneck capacity	4 Mbps	Without cross-traffic
Cross-traffic	true, false	Additional competing traffic
Effective bandwidth	b, $10b$	$\times 10$ with cross-traffic
C/BDP	[1.0, 1.2, 1.5, 3.0]	Buffer size relative to BDP

approach that uses a baseline buffer-based ABR method [17] at the same decision epoch. After each adaptation, the stored notification is cleared and t_{last} is updated, initiating the next cooldown interval (lines 15–16). By responding directly to network feedback rather than relying solely on historical throughput estimates, the approach is intended to reduce stalls and maintain stable quality under variable network conditions.

C. Simulation Setup and Experimental Design

To evaluate the proposed adaptation approach, we simulate a video client and server connected through a router in a single-bottleneck topology with constant-capacity and error-free links. A custom Python simulator models the interactions among network nodes, streaming components, and background traffic. We consider TCP with segmented packet transmission and a maximum packet size of 1294 B. The nodes implement NewReno-style congestion control, including slow start, Additive Increase/Multiplicative Decrease (AIMD), and fast recovery [18], while mechanisms such as delayed acknowledgments and pacing are not included. Although NewReno provides a canonical AIMD baseline, the simulator is modular and can be extended with CUBIC [19] and QUIC to assess the generality of the results. The bottleneck router uses Deficit Round Robin for fair scheduling, while we use DROPTAIL with a buffer size of 1.2 times the bandwidth–delay product (BDP) as baseline queuing discipline. Signaling between the router and client is assumed to be instantaneous and error-free.

1) Simulation Setup: The simulation comprises video generation, adaptive streaming, and cross-traffic generation. Video segments are produced using a stochastic, data-driven model fitted to movie traces following [20]. The model supports arbitrary group-of-pictures structures and four quality levels: 1080p ($\approx$ 5 Mbps), 720p, 480p, and 360p, with the lower resolutions obtained through linear scaling. The streaming actor executes parallel processes for segment downloading, virtual playback, and quality adaptation. Playback stalls when the buffer is empty and resumes after a predefined buffer threshold is reached. The baseline ABR scheme follows the buffer-based method of [17], which uses a staircase function to map buffer occupancy within a reservoir-cushion model to a quality level. The logged quality indices $q = 1, 2, 3, 4$ correspond to 360p, 480p, 720p, and 1080p, respectively. For the cross-traffic configuration, we activate two independent generators. The *Steady* generator initiates 20 MB downloads with exponentially distributed interarrival times of 10 s on average. The *Bursty* generator uses a mean interarrival time of 1 s and initiates 12–28 parallel transfers with a mean file size of 100 KB. Each generator produces an average offered load of 16 Mbps, yielding a combined mean load of 32 Mbps.

Table II: AQM-specific parameters without EBN.

AQM	Parameter	Values	Notes
CoDel	Target delay [ms]	2, 5, 10	Default [6]: 5 ms
	Interval [ms]	100, 150, 200	Default [6]: 100 ms
PIE	Queue delay target [ms]	10, 15, 20	Default [5]:15 ms
	Max burst [ms]	100, 150, 200	Default [5]:150 ms
MORAIS	Min threshold ($\min_{th}$) [ms]	20, 30, 50	Default setup not
	Max threshold ($\max_{th}$) [ms]	100, 150, 200	available in [9]
ARED	Min threshold ($\min_{th}$)	default, $\frac{1}{6}C$	As in [4], [21]
	Max threshold ($\max_{th}$)	$3{\cdot}\min_{th}$, C	As in [4], [21]
	Max probab. ($\max_P$)	0.02, 0.04	As in [4], [21]

Table III: EBN-related parameters.

Parameter	Values
Router EBN scheme	None, Alg. 1
Client ABR scheme	Buffer-based [17], Alg. 2, Alg. 2 (Hybrid)
Cooldown (Δt)	3 s
Min threshold ($\theta_{\min}$)	0.3

2) Experimental Design: The simulation parameters fall into three groups: general, AQM-specific, and EBN-related. Table I summarizes the general parameters used independently of the selected AQM. Each run lasts 600 s, reducing the influence of initial transients while capturing stochastic variation in the AQM and cross-traffic processes. The router delay is set to 20 ms to represent a typical Internet path [22], and the target playback buffer is 20 s, following a common YouTube client configuration. We select a base link capacity of 4 Mbps, slightly below the 5 Mbps required for 1080p streaming in our setting, to induce quality adaptation and occasional stalls. We evaluate scenarios both with and without cross-traffic: in the former case, the link capacity is increased to 40 Mbps to accommodate multiple competing flows. The router buffer capacity C is expressed as a multiple of the BDP, representing different levels of buffer provisioning and potential bufferbloat [3]. Table II lists the AQM-specific parameters. For CoDel, PIE, and MORAIS, we evaluate three values per parameter, including the defaults for CoDel and PIE. No default MORAIS configuration is provided in [9]. For ARED, we consider both its default values and the alternatives proposed by Akhtar et al. [21]. The EBN-related parameters are summarized in Table III. Based on preliminary testing, we set the router-side threshold to $\theta_{\min} = 0.3$ and the client-side cooldown interval to $\Delta t = 3$ s. For each parameter configuration, we conduct 30 independent simulations, resulting in nearly 60,000 runs.

IV. EVALUATION

In this section, we evaluate the impact of different AQM schemes on video streaming traffic using default parameters, compare their performance across alternative settings, and assess the performance of the proposed EBN approach.

A. Comparative AQM Analysis

The first set of experiments compares ARED, PIE, CoDel, and MORAIS with DROPTAIL for HAS traffic using the

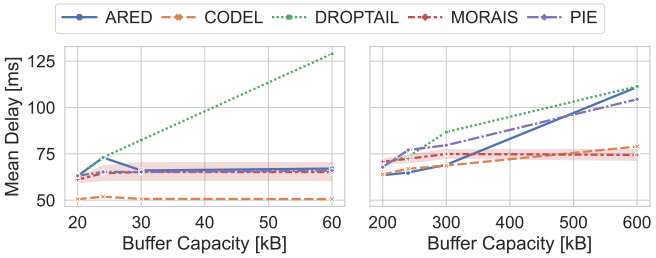

Figure 1: Packet delay: crosstraffic off (left) vs. on (right).

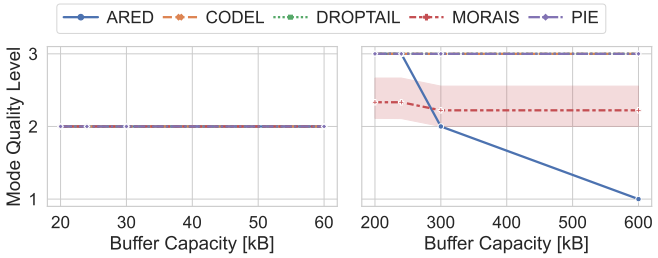

Figure 2: Video quality: crosstraffic off (left) vs. on (right).

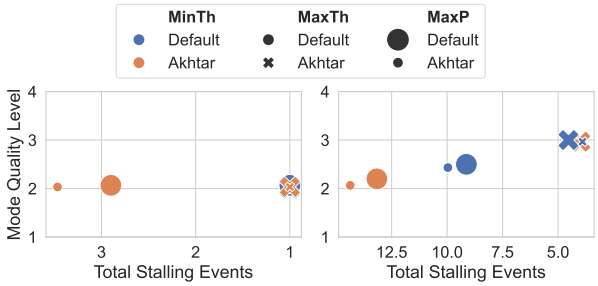

Figure 3: ARED, quality: crosstraffic off (left) vs. on (right).

default parameters listed in Table II. We evaluate throughput, mean packet delay, jitter, packet-drop rate, and modal playback quality, focusing on latency and video quality to illustrate the main trade-offs among the queue-management schemes.

Throughput: Without cross-traffic, DROPTAIL achieves nearly full link utilization (∼4 Mbps), whereas CoDel yields the lowest throughput (< 3 Mbps), consistent with its more aggressive packet dropping. With cross-traffic, PIE and CoDel attain throughput comparable to DROPTAIL (∼5 Mbps), while ARED decreases to approximately 1 Mbps for large buffers. MORAIS maintains a relatively stable throughput of 3–4 Mbps across buffer sizes in both scenarios.

Jitter and Drop Rate: Jitter and packet loss are related because losses activate TCP congestion control and can increase rate fluctuations. Without cross-traffic, the AQM schemes maintain jitter between 1 and 1.3 ms, whereas DROPTAIL exhibits lower jitter at larger buffer sizes, coinciding with fewer packet drops. With cross-traffic, ARED reaches the highest jitter and drop rate, peaking at 15 ms and 2.5 %, respectively. CoDel and PIE keep jitter below 2 ms and drop rates near 0.4 %, while MORAIS exhibits slightly higher but relatively consistent drop rates.

Average Packet Delay: Figure 1 compares the mean delay of HAS packets across buffer capacities. Colors identify the queue-management schemes, while the shaded region for MORAIS covers the full parameter range in Table II. The delay under DROPTAIL increases approximately linearly with buffer capacity because of persistent queuing. Without cross-traffic (left), the AQM schemes maintain nearly constant delays. CoDel achieves the lowest value of approximately 50 ms, at least 10 ms below MORAIS, ARED, and PIE. Cross-traffic increases delay because of greater contention. ARED then approaches DROPTAIL, rising from below 70 ms to approximately 110 ms as the buffer grows, while PIE follows a similar but slightly lower trend. CoDel increases more gradually and remains below 80 ms, whereas MORAIS is the only scheme that maintains a relatively stable delay of approximately 70 ms.

Application-Layer Quality: Turning to application-level performance, Figure 2 presents the modal video quality across buffer capacities, where levels 1–4 correspond to 360p, 480p, 720p, and 1080p, respectively. Without cross-traffic and at a link capacity of 4 Mbps, all schemes deliver 480p. With cross-traffic and a capacity of 40 Mbps, DROPTAIL, PIE, and CoDel remain stable, whereas the aggregated modal quality for MORAIS varies between 2.0 and 2.6. ARED is particularly sensitive to buffer capacity: its modal quality decreases from 720p to 360p as C increases from 240 kB to 600 kB.

AQM-specific Parameter Study

We next examine how AQM parameter tuning affects network and streaming performance. CoDel and PIE were less sensitive to parameter changes. For CoDel, an $interval$ of 200 ms combined with the default $target$ minimized performance variation, while the default PIE configuration provided stable video quality. We therefore focus on ARED and MORAIS, using $C = 24$ kB without cross-traffic and $C = 240$ kB with cross-traffic. In Figures 3, 4, 5, and 6, preferable operating points are located in the upper-right quadrant.

ARED Parameter Sensitivity: The results reveal a trade-off between throughput and latency. Without cross-traffic, most configurations sustain approximately 4 Mbps. Combining $min_{th} = \frac{1}{6}C$ [21] with the default max_{th} reduces delay by approximately 10 ms at a small throughput cost. With cross-traffic, setting $max_{th} = C$ maximizes throughput but increases delay, whereas the default configuration provides the lowest latency. Thus, the default parameters favor delay-sensitive streaming, while the configuration proposed by Akhtar et al. [21] favors throughput. Figure 3 illustrates the corresponding trade-off between stalling frequency and video quality. Marker color, shape, and size represent min_{th}, max_{th}, and max_P, respectively. For example, a small blue circle denotes default values for min_{th} and max_{th} with the max_P value proposed by Akhtar [21]. With cross-traffic (right), performance is primarily influenced by max_{th}, while variations in min_{th} and max_P have smaller effects. The max_{th} setting from [21] yields higher modal quality (720p) and fewer stalls than the default setting, as indicated by the crosses in the upper-right region of the right-hand plot. Figure 4 further examines stalling severity in terms of event frequency and

Table IV: Experimental configurations used in Figure 7.

Scenario	C1	C2	C3	C4	C5	C6
AQM	ARED: min_{th}, max_{th} as in [21]	MORAIS: $min_{th} = 50$, $max_{th} = 100$	ARED	ARED	MORAIS: $min_{th} = 20$, $max_{th} = 100$	MORAIS: $min_{th} = 20$, $max_{th} = 100$
EBN Algorithm	None	None	1, 2	1, 2 (Hyb.)	1, 2	1, 2 (Hyb.)

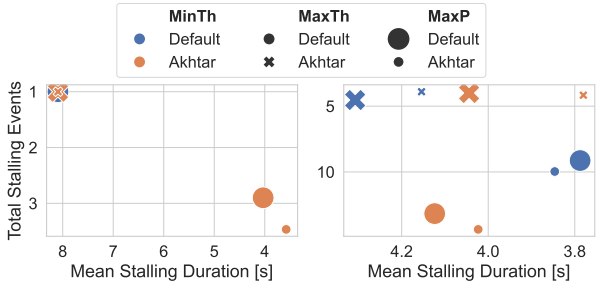

Figure 4: ARED, stalls: crosstraffic off (left) vs. on (right).

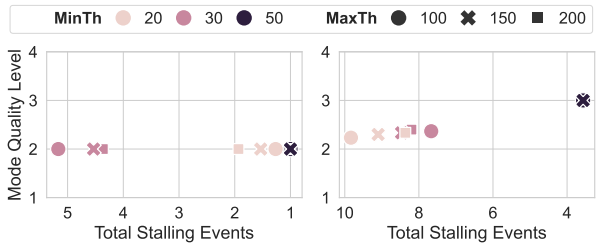

Figure 5: MORAIS, quality: crosstraffic off (left) vs. on (right).

mean duration. Setting $min_{th} = \frac{1}{6}C$ generally produces fewer and shorter stalls, particularly with cross-traffic.

MORAIS Parameter Sensitivity: For MORAIS, the throughput-delay trade-off is mainly governed by min_{th}. With or without cross-traffic, setting $min_{th} = 50$ ms yields throughput close to 4 Mbps but increases delay. Values of $min_{th} \leq 30$ ms reduce latency by more than 10 ms in some configurations, at a throughput cost of up to 1 Mbps. At $min_{th} = 50$ ms, max_{th} has little effect, possibly because more packets are enqueued directly rather than subjected to probabilistic dropping. Figure 5 shows the effects of these parameters on video quality and stalling. Without cross-traffic, all configurations maintain 480p on the 4 Mbps link. However, $min_{th} = 30$ ms produces more than four stalls on average, whereas settings of 20 ms and 50 ms result in fewer than two. With cross-traffic, $min_{th} = 50$ ms increases the modal quality by nearly one level while maintaining fewer than four stalls. By comparison, max_{th} has negligible influence. Figure 6 shows that $min_{th} = 50$ ms produces fewer but longer stalls, whereas $min_{th} \leq 30$ ms results in shorter but more frequent stalls. This pattern occurs both with and without cross-traffic, although it is less pronounced in the latter case.

Summary, Takeaways, and Answer to RQ1: The results reveal trade-offs among throughput, latency, transmission efficiency, and playback resilience. DROPTAIL achieves the highest throughput but causes bufferbloat and high delay. Without cross-traffic, CoDel provides the lowest latency (approximately 50 ms) at the cost of lower throughput and more frequent drops. With cross-traffic, MORAIS maintains stable delay, whereas PIE and ARED approach DROPTAIL at large buffers. Most schemes sustain HD quality, although MORAIS exhibits quality fluctuations and ARED can drop to 360p. CoDel combines stable HD playback with low latency. Parameter tuning is most beneficial for ARED and MORAIS, while CoDel and PIE are less parameter-sensitive. For ARED, max_{th} is the main factor: the default favors low latency, whereas the configuration of Akhtar [21] better balances stalling and quality. For MORAIS, min_{th} governs the throughput-delay trade-off: higher values increase throughput and reduce stall frequency but prolong stalls, while $min_{\text{th}} = 20$ ms provides a balanced compromise.

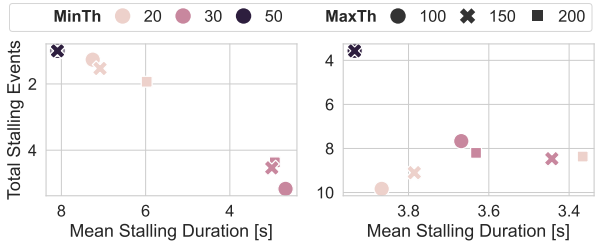

Figure 6: MORAIS, stalls: crosstraffic off (left) vs. on (right).

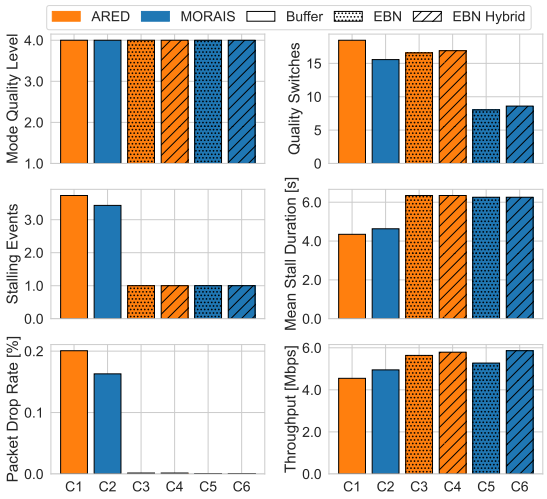

Figure 7: Best-performing configurations, with cross-traffic.

B. Evaluation of EBN-based Adaptation

The results indicate that EBN can improve HAS playback stability when appropriately integrated with the underlying AQM logic. With ARED and MORAIS, EBN maintains high playback quality and buffer occupancy, whereas with CoDel and PIE it triggers aggressive upshifts followed by buffer depletion and frequent stalls. We therefore focus on EBN combined with ARED and MORAIS. Table IV

summarizes the best-performing configurations among those evaluated. Six scenarios are considered: two without EBN (C1, C=300 kB; C2, C=200 kB), two using Algorithm 2 (C3: ARED, C=600 kB; C5: MORAIS, C=240 kB), and two using its hybrid variant (C4: ARED, C=600 kB; C6: MORAIS, C=240 kB). Cross-traffic is active in all scenarios, with $\Delta t = 3\,\mathrm{s}$ and $\theta_{\min} = 0.3$. Figure 7 shows that all configurations achieve the highest feasible video quality. In the upper-right plot, MORAIS produces fewer quality switches with both EBN variants, reducing adaptation frequency. The middle-left plot shows that EBN reduces the mean number of stalls to approximately one across both AQMs, corresponding to the initial buffering phase. Accordingly, the middle-right plot reports longer mean stall durations with EBN because this initial buffering period is longer, whereas the baselines experience multiple stalls during playback. The largest improvement appears in the lower-left plot, which reports the packet-drop rate of HAS flows. Without EBN, drop rates reach 0.20 % for ARED and 0.16 % for MORAIS. With EBN, they approach zero because AQM marks generate feedback rather than packet drops, except when the queue is full. The reduction in losses is also accompanied by higher throughput, as shown in the lower-right plot. With EBN, MORAIS achieves comparable performance with smaller buffers than ARED, while the ARED setup in [21] remains preferable with conventional ABR.

Summary, Takeaways and Answer to RQ2: Suitably configured AQM-based EBN enables proactive adaptation and improves playback resilience, resulting in fewer quality switches, no stalls after initial buffering, near-zero HAS packet drops, and higher throughput. The reduction in packet loss also indicates improved transmission efficiency. Effectiveness depends on the underlying AQM: EBN integrates well with ARED and MORAIS, whereas CoDel and PIE trigger overly aggressive quality increases that destabilize playback. In practice, EBN can map AQM marking decisions at the bottleneck router to lightweight, client-visible signals, such as ECN marks or control-plane notifications, combined with ABR logic that prioritizes buffer stability. A hybrid design can improve robustness by reverting to conventional ABR decisions when EBN feedback is unavailable or unreliable. These results also motivate extending the three-state feedback to a continuous, smoothed signal based on per-segment ECN-marking density, consistent with the frequent marking used by L4S.

V. Conclusions

This work examines the interaction between AQM and HAS and introduces a network-assisted ABR mechanism for resilient, congestion-aware playback. AQM selection and configuration affect network-level QoS, playback resilience, and transmission efficiency. With default settings, CoDel and PIE are less parameter-sensitive, whereas ARED and MORAIS benefit more from tuning. Compared with the evaluated conventional approaches, EBN reduces stalling events by up to 70 %, quality switches by up to 49 %, and HAS packet drops by approximately 99 %, while maintaining comparable video quality. These reductions improve playback continuity and indicate lower loss-related transmission overhead. The evaluation is simulation-based and uses a simplified TCP NewReno model. Future work should validate the findings through ns-3 simulations, packet-level emulation, and testbed experiments, including CUBIC/BBR congestion control and QUIC-based transport. EBN could also be extended from three states to a continuous, smoothed measure of per-segment ECN-marking density. Integration with L4S/DualQ [14, 15] could provide finer-grained ABR feedback through more frequent ECN marking.

Acknowledgments

Partly funded by the German Federal Ministry of Research, Technology and Space in Sustainet–Advance Grant 16KIS2282 (Univ. Würzburg) and NextGenerationEU's program "*MUR - Fondo Promozione e Sviluppo - DM737/2021*".

References

[1] Sandvine, "The Global Internet Phenomena Report," March 2024, accessed 26.01.2026. [Online]. Available: https://www.applogicnetworks.com/phenomena

[2] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Communications of the ACM*, 2012.

[3] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *Transactions on Networking*, 2002.

[4] S. Floyd *et al.*, "Adaptive RED: An Algorithm for Increasing the Robustness of RED's Active Queue Management," 2001.

[5] "Proportional Integral Controller Enhanced (PIE): A Lightweight Control Scheme to Address the Bufferbloat Problem," accessed 20.11.2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8033

[6] "Controlled Delay Active Queue Management," accessed 20.11.2025. [Online]. Available: https://www.rfc-editor.org/rfc/rfc8289

[7] J. Kua *et al.*, "The Impact of Active Queue Management on DASH-based Content Delivery," in *Local Computer Networks*. IEEE, 2016.

[8] J. I. Sandoval *et al.*, "QUIC Adaptive Video Streaming Performance with 5G RAN Queue Management," in *International Mediterranean Conference on Communications and Networking*. IEEE, 2025.

[9] W. G. de Morais *et al.*, "Application of Active Queue Management for Real-Time Adaptive Video Streaming," *Telecom. Systems*, 2022.

[10] C. E. M. Santos *et al.*, "Improving Perceived Quality of Live Adaptative Video Streaming," *Entropy*, 2021.

[11] J. Kua, P. Branch, and G. Armitage, "Detecting Bottleneck Use of PIE or FQ-CoDel Active Queue Management during DASH-like Content Streaming," in *Conference on Local Computer Networks*. IEEE, 2020.

[12] R. Bhattacharyya *et al.*, "QFlow: A Learning Approach to High QoE Video Streaming at the Wireless Edge," *Transactions on Netw.*, 2021.

[13] P.-L. Ageneau and G. Armitage, "Network Delivery Time Control: a Novel Approach to Rate Adaptation for Low-Latency Video Flows," in *Conference on Local Computer Networks*. IEEE, 2025, pp. 1–7.

[14] B. Briscoe *et al.*, "Low Latency, low Loss, Scalable Throughput (L4S) Internet Service: Architecture," in *RFC 9330*, 2023.

[15] K. De Schepper and G. White, "RFC 9332: Dual-Queue Coupled Active Queue Management (AQM) for Low Latency, Low Loss, and Scalable Throughput (L4S)," 2023.

[16] D. Tsilimantos, T. Karagkioules, and S. Valentin, "Classifying Flows and Buffer State for YouTube's HTTP Adaptive Streaming Service in Mobile Networks," in *Multimedia Systems Conference*, 2018.

[17] T.-Y. Huang *et al.*, "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *ACM SIGCOMM Conference*, 2014.

[18] M. Allman *et al.*, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[19] L. Xu, S. Ha, I. Rhee, V. Goel, and L. Eggert, "CUBIC for Fast and Long-Distance Networks," *RFC 9438*, 2023.

[20] U. K. Sarkar *et al.*, "Modeling Full-Length Video using Markov-Modulated Gamma-based Framework," *Trans. on Networking*, 2003.

[21] S. Akhtar, A. Francini, D. Robinson, and R. Sharpe, "Interaction of AQM Schemes and Adaptive Streaming with Internet Traffic on Access Networks," in *Global Communications Conference*. IEEE, 2014.

[22] F. Loh *et al.*, "QoS and QoE Study of the European 5G Mobile Networks for Next Generation of Applications," *IEEE Comm. Magazine*, 2025.