Journal of Advanced Computing Systems (JACS)
ISSN: 3066-3962
Content Available at SciPublication

QoE-Driven Reinforcement Learning for Joint Bitrate, Rebuffering, and TTFF Optimization in HLS/DASH

Eric Wang[1], *Heyu Wang*[2]
[1]*Computer Science, UCLA, CA, USA*
[2]*Electrical and Computer Engineering, Rice University, TX, USA*
eric.wang03415@gmail.com

DOI: 10.69987/JACS.2023.30204

Keywords

adaptive bitrate streaming; QoE; reinforcement learning; HLS; MPEG-DASH

Abstract

HTTP adaptive streaming over HLS/DASH must balance delivered visual quality against playback interruptions, bitrate variation, and startup delay. In many deployed players, time-to-first-frame (TTFF) is still handled through startup heuristics rather than being optimized jointly with steady-state adaptive bitrate (ABR) decisions. This paper studies a trace-driven controller family that combines a PPO+GAE actor-critic policy with two deployment-oriented constraints: a safety supervisor that caps bitrate by an online throughput estimate and an optional startup cap that operates only before playback begins. We evaluate the controller family on 40 mobile HSDPA throughput traces from MMSys'13 using a simulator with 2 s segments, a 6-level bitrate ladder, and a unified QoE metric that rewards bitrate and penalizes rebuffering, bitrate changes, and TTFF. In the four-way controller comparison on the held-out 8-trace test split, the 750 kbps startup-cap operating point (SafeRL-TTFF-750) achieves the highest mean QoE (136.125 ± 58.994), improves mean TTFF by 16.6% relative to the throughput-based RB baseline, and keeps mean rebuffering at 0.228 ± 0.556 s. On the full 40-trace set, SafeRL-TTFF-750 and RB are effectively tied in mean QoE, with the former trading slightly higher bitrate and lower TTFF for higher rebuffering. An ablation study shows that the safety supervisor is essential, and that stricter startup caps can reduce TTFF further with only small changes in scalar QoE. The results support a practical conclusion: learned ABR can be useful on mobile traces when RL decisions are wrapped in transparent safety and startup controls.

1. Introduction

HTTP-based adaptive streaming (HAS) remains the dominant architecture for large-scale Internet video delivery because it reuses standard HTTP/CDN infrastructure while allowing the player to adapt representation bitrate to observed network conditions [4]-[6]. In practice, HAS is typically realized through HTTP Live Streaming (HLS) and MPEG-DASH, which expose a ladder of pre-encoded bitrates and let the client choose one representation per segment request.

ABR control is difficult because the client observes path capacity only indirectly, through segment download times shaped by transport dynamics, server behavior, and wireless scheduling. Mobile networks make the problem harder: throughput can vary at timescales comparable to the segment duration, so both prediction and reactive control are noisy. Prior QoE research therefore emphasizes a familiar triad of objectives: maximize delivered quality, minimize rebuffering, and limit unnecessary bitrate switching. Rebuffering is especially important because even modest increases in buffering time can have a disproportionate effect on user engagement [9].

TTFF, also called startup delay or join time, is a related but distinct user-facing metric. It measures the interval between the user's play request and the first rendered frame. In many practical players, TTFF is handled by heuristics such as starting at a low bitrate and waiting for a small startup buffer before rendering. Those heuristics are often effective, but they are rarely optimized in a way that is fully consistent with the steady-state bitrate-versus-stall tradeoff [23].

Learning-based ABR offers an appealing alternative because it can optimize an end-to-end QoE objective directly rather than relying on a fixed analytic control law. Pensieve showed that reinforcement learning can learn competitive ABR policies from trace-driven simulation [15]. PPO and GAE further improved the stability of policy-gradient learning by combining clipped policy updates with lower-variance advantage estimates [16], [17]. At the same time, practical deployment experience suggests that unconstrained learned policies remain brittle under distribution shift and therefore benefit from explicit safeguards [24].

This paper revisits that deployment question on real mobile throughput traces. Rather than asking whether an unconstrained RL policy can replace conventional ABR logic, we study a more conservative controller family in which a learned actor-critic bitrate policy is wrapped by an online safety supervisor and, when desired, by a startup-only bitrate cap. The safety layer protects against gross overshoot under mobile variability, while the startup layer exposes TTFF as an explicit operating-point choice instead of a hidden side effect [25].

The revised manuscript makes three contributions. First, it formulates a TTFF-aware ABR controller family in which a PPO-trained actor-critic policy is constrained by an online safety supervisor and optional pre-playback startup caps. Second, it evaluates that family against throughput-based, buffer-based, and short-horizon planning baselines on 40 real mobile throughput traces from MMSys'13 using a unified QoE metric. Third, it separates two questions that were conflated in the previous draft: which controller family performs best on the scalar QoE objective, and which startup-cap operating point offers the most desirable startup behavior [26].

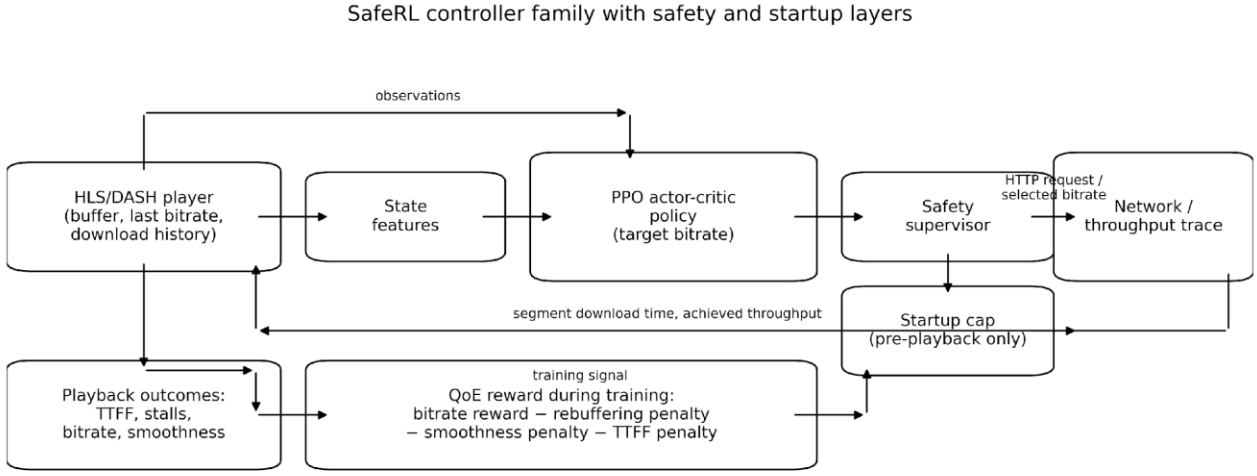

Fig. 1. SafeRL controller family with safety and startup layers.

2. Methods

2.1 Trace source and selection

We model segment-based adaptive streaming as a sequential decision problem. At each segment request, the player chooses one representation from a discrete bitrate ladder, downloads the selected segment over a time-varying link, updates its playback buffer, and receives a QoE-based reward. The evaluation uses the publicly available MMSys'13 HSDPA bandwidth archive, which contains application-layer logs from adaptive HTTP streaming sessions collected in Telenor's 3G/HSDPA network in Norway [1], [2]. The archive notes that the adaptive streams were downloaded at maximum speed, that the segment duration was fixed to 2 seconds, and that the measurement period spans 2010-09-13 to 2011-04-21 [1].

We use 40 traces spanning eight route categories: bus, metro, ferry, three tram categories, car, and train. Table 1 summarizes the selected subset, and Appendix A lists the filenames explicitly for reproducibility.

Table 1. Dataset summary for the 40 selected MMSys'13 HSDPA traces (per route).

Route	Traces	Duration median (s)	Duration p10 (s)	Duration p90 (s)	Avg throughput median (kbps)	Avg throughput p10 (kbps)	Avg throughput p90 (kbps)
Bus	7	787.7	528.7	1195.8	2259.1	1668.5	2557.0

Car	7	437.1	421.6	9329.3	1715.3	564.1	1851.1
Ferry	4	1183.0	1102.8	1276.1	1375.1	1219.1	1546.5
Metro	7	871.0	456.4	1095.9	571.4	410.6	1019.6
Train	5	2200.8	2108.8	2601.7	1069.0	953.7	1237.8
Tram 1	5	1322.7	1284.4	1411.6	786.3	742.9	887.1
Tram 2	4	1284.9	1215.4	1447.1	709.3	672.7	741.2
Tram 3	1	1509.5	1509.5	1509.5	798.6	798.6	798.6

2.2 Trace processing and playback model

Each log sample provides the number of bytes received since the previous measurement and the elapsed time in milliseconds [1]. We convert each sample to instantaneous application-layer throughput as throughput_kbps = 8 × bytes / ms and interpret the value as piecewise constant over that sample's duration. Because the sampling interval varies across lines, download-time computation integrates the trace in continuous time: if the remaining segment bits exceed the capacity of the current sample, the simulator consumes the full sample and advances; otherwise, it consumes only the fraction needed to complete the segment.

The player requests segments sequentially. Each request incurs a fixed 100 ms overhead intended to capture round-trip and request-processing latency. Before playback starts, downloaded segments accumulate in the startup buffer without being consumed. Playback begins when the buffer reaches 4 seconds, corresponding to two 2-second segments. TTFF is defined as the wall-clock time required to reach that threshold. After startup, the playback buffer drains during segment download. If the buffer reaches zero before the segment completes, the excess time is counted as rebuffering.

All methods are evaluated on an identical content length of 120 segments (240 seconds of media). Because the real traces have finite duration, the simulator loops the throughput trace when it ends, following standard practice in trace-driven ABR evaluation [15].

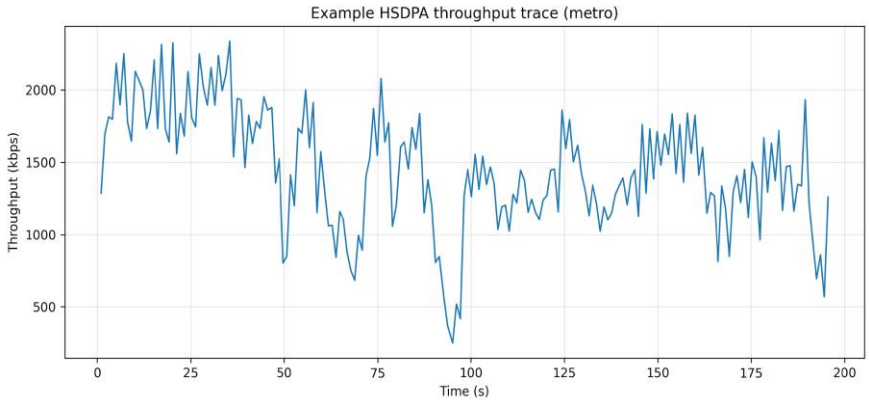

Fig. 2. Example HSDPA throughput trace from the MMSys'13 dataset (metro route).

2.3 Video ladder and QoE objective

We use a six-level constant-bitrate ladder at 300, 750, 1200, 1850, 2850, and 4300 kbps. With 2-second segments, this corresponds to segment sizes from 75 kB to 1075 kB. Using a fixed ladder isolates adaptation behavior from content-dependent segment-size variability.

The session QoE objective rewards delivered bitrate while penalizing rebuffering, switching, and TTFF. Let q t denote the selected bitrate for segment t in Mbps, R the total rebuffering time in seconds, T the TTFF in seconds, and |q_t − q_{t−1}| the absolute bitrate change between adjacent segments. The session-level objective is:

$$QoE = \Sigma\_t\, q\_t - \alpha R - \beta \Sigma\_t\, |q\_t - q\_\{t-1\}| - \gamma T.$$

We use α = 4.3 and β = 1.0, following the Pensieve-style scalar QoE setting [15], and set γ = 0.5 so that startup delay matters without overtaking rebuffering as the dominant penalty. During RL training, the TTFF term is implemented densely as a time cost applied only before playback starts; this makes the cumulative pre-playback penalty equal to γT while providing more immediate learning signal.

Table 2. Video representation ladder used in the experiments (2 s segments).

Representation	Bitrate (kbps)	Segment duration (s)	Segment size (kB)
R1	300	2.0	75.0
R2	750	2.0	187.5
R3	1200	2.0	300.0
R4	1850	2.0	462.5
R5	2850	2.0	712.5
R6	4300	2.0	1075.0

Table 3. QoE model terms and weights used for training and evaluation.

Term	Symbol	Definition	Weight
Video quality	q_t	Selected bitrate for segment t (Mbps)	1.0
Rebuffering	r_t	Playback stall time incurred while downloading segment t (s)	4.3
Smoothness	$\|\Delta q_t\|$	Absolute bitrate change between consecutive segments (Mbps)	1.0
Startup delay (TTFF)	TTFF	Time from session start until playback starts (s)	0.5

2.4 Controllers and training configuration

We compare three baseline ABR families and one SafeRL operating point. RB is a throughput-based baseline that estimates throughput as the harmonic mean of the last five measured segment throughputs and chooses the highest representation below 0.85 times that estimate. BBA is a buffer-based controller with a 4-second reservoir and a 10-second cushion. MPC-3 predicts future throughput as 0.9 times the harmonic mean of the last five throughput observations and performs brute-force planning over a horizon of three segments.

The SafeRL policy is an actor-critic multilayer perceptron with two hidden layers of 64 units and tanh activations. Its state vector concatenates the last five measured throughputs, the last five download times, the current buffer level, the last selected bitrate, a binary playback-started flag, and the remaining startup-buffer budget. The action space is the six representation indices.

Training uses PPO with GAE, discount factor 0.99, GAE parameter 0.95, and clip parameter 0.2 [16], [17]. To reduce unsafe early exploration, the policy is first initialized by behavior cloning from BBA on 4000 state-action pairs and then fine-tuned with six PPO updates of 256 environment steps each. The reported train/test split uses 32 training traces and 8 held-out test traces under a fixed random split.

At inference time, the learned policy can be wrapped by two deployment layers. The safety supervisor computes the harmonic mean of the last five measured throughputs and caps the selected bitrate so that it does not exceed the current estimate. The optional startup cap further limits bitrate only until playback begins. In the main four-way comparison, the reported operating point is SafeRL-TTFF-750, which applies a 750 kbps startup cap. Table 9 later compares this setting to safety-only SafeRL and to a stricter 300 kbps startup cap.

Table 4. ABR controller configurations and key hyperparameters.

Method	Key settings
RB (throughput-based)	Harmonic-mean throughput over last 5 segments; safety=0.85; choose max bitrate ≤ estimate
BBA (buffer-based)	Reservoir=4 s; cushion=10 s; linear mapping buffer→bitrate

MPC-3	Horizon=3 segments; predicted throughput = harmonic mean of last 5 ×0.9; brute-force search over 6^3 sequences
SafeRL-TTFF-750	Actor-critic policy (2×64 MLP) trained with PPO+GAE; safety filter caps bitrate to throughput estimate; startup cap: bitrate ≤ 750 kbps until playback starts

3. Results and Discussion

Table 1 shows strong heterogeneity across the selected routes. Median average throughput is highest for the bus traces (2.259 Mbps) and lowest for the metro traces (0.571 Mbps). The train traces are the longest sessions, while the car category includes both short urban drives and a much longer trip, which explains its wide duration spread. Figure 2 illustrates the behavior of one metro trace: throughput varies rapidly at short timescales, but the more consequential challenge for ABR control is the occurrence of sustained low-rate intervals that can drain the playback buffer if the controller over-commits.

Table 5 and Figures 3-4 summarize the main held-out comparison over the eight test traces. In this four-way family comparison, SafeRL-TTFF-750 attains the highest mean QoE (136.125 ± 58.994). Relative to RB, it improves mean QoE by 6.4%, increases mean delivered bitrate by 5.0%, and reduces mean TTFF from 2.221 s to 1.852 s (16.6%) while keeping mean rebuffering at 0.228 s. It also achieves the lowest mean smoothness penalty on the test split (51.943 kbps), suggesting that the learned policy plus safety cap makes smaller corrections once playback begins.

Table 5. Held-out test-set results (mean ± std across 8 traces).

Method	QoE	Avg bitrate (kbps)	Rebuffering (s)	TTFF (s)	Smoothness (kbps)	SRR
RB	127.981 ± 67.594	1144.323 ± 601.050	0.025 ± 0.072	2.221 ± 0.265	68.225 ± 42.101	0.000 ± 0.000
BBA	124.135 ± 60.252	1410.417 ± 632.476	2.770 ± 6.455	0.933 ± 0.212	275.105 ± 102.722	0.011 ± 0.025
MPC-3	110.674 ± 59.783	1529.896 ± 745.168	11.512 ± 7.623	2.534 ± 0.583	186.082 ± 85.589	0.045 ± 0.029
SafeRL-TTFF-750	136.125 ± 58.994	1201.771 ± 483.811	0.228 ± 0.556	1.852 ± 0.350	51.943 ± 20.795	0.001 ± 0.002

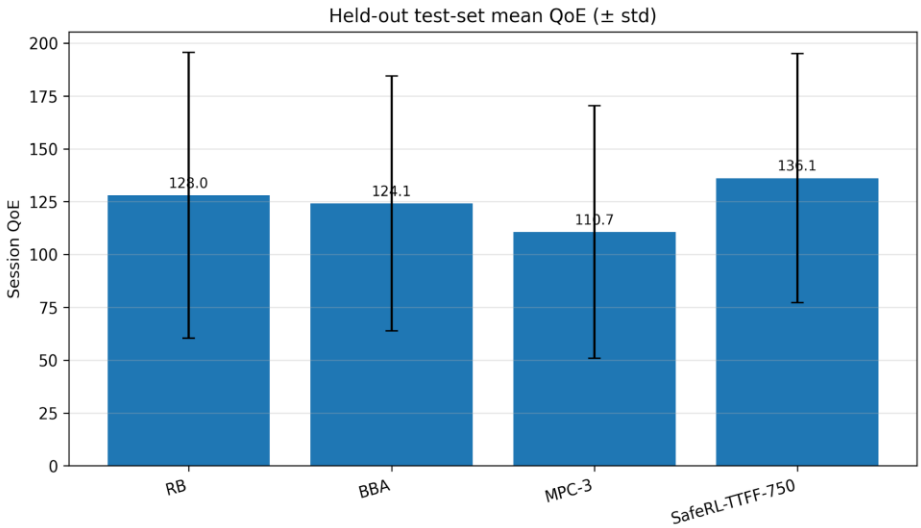

Fig. 3. Held-out test-set mean QoE with one-standard-deviation error bars.

BBA shows the startup-versus-stability tradeoff from the opposite direction. It reaches playback fastest (TTFF 0.933 s), but that gain comes with much larger bitrate oscillations and more rebuffering, which lowers scalar QoE under the chosen reward weights. MPC-3 achieves the highest mean bitrate among the baselines, but the improvement is overwhelmed by rebuffering under throughput-prediction error. The result is consistent with the view that planning-based ABR can be appealing when prediction is accurate yet fragile on highly variable mobile links.

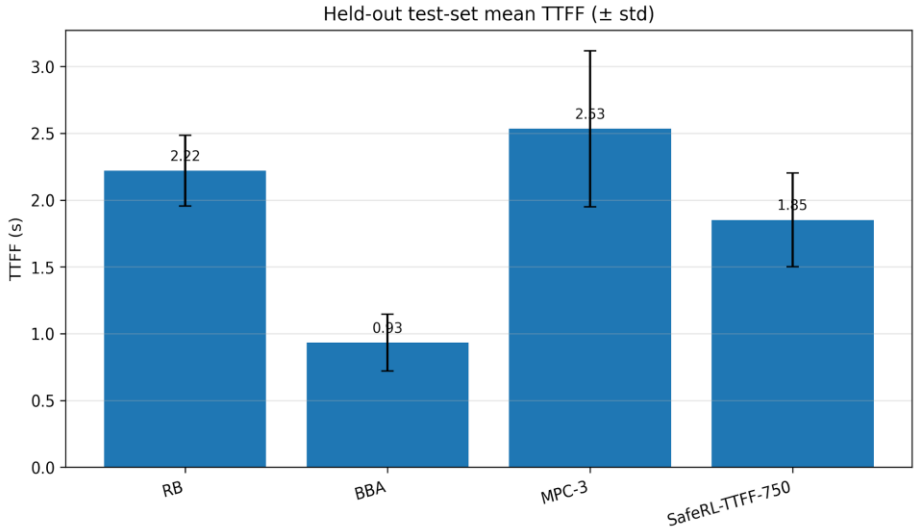

Fig. 4. Held-out test-set mean TTFF with one-standard-deviation error bars.

On the full 40-trace set (Table 6), SafeRL-TTFF-750 and RB are effectively near tied on mean QoE: 100.957 versus 100.515. The learned controller offers a higher mean bitrate (1138.760 vs. 1027.812 kbps) and a shorter mean TTFF (3.238 vs. 3.622 s), but it pays for those gains with higher mean rebuffering (6.224 vs. 3.045 s). This full-set result is therefore more nuanced than the held-out ranking, and the revised manuscript treats the advantage as marginal rather than as clear dominance.

Table 7 and Figure 5 show that route-level behavior is not uniform. SafeRL-TTFF-750 outperforms RB on car, train, and all three tram categories, but RB remains stronger on bus, ferry, and metro. The previous draft overstated the bus-route behavior; the corrected interpretation is that the learned controller helps most in moderate conditions where extra bitrate can be exploited without frequent stalls, while RB remains difficult to beat on conservative or harsh low-throughput routes. This route dependence also explains why the full-set mean QoE difference between the two methods is small.

Table 6. Full-set results (mean ± std across 40 traces).

Method	QoE	Avg bitrate (kbps)	Rebuffering (s)	TTFF (s)	Smoothness (kbps)	SRR
RB	100.515 ± 74.163	1027.812 ± 448.269	3.045 ± 9.320	3.622 ± 5.929	66.534 ± 30.899	0.011 ± 0.033
BBA	69.134 ± 139.944	1322.031 ± 486.284	14.249 ± 28.168	1.428 ± 1.857	231.292 ± 68.964	0.047 ± 0.085
MPC-3	31.188 ± 236.010	1422.802 ± 540.593	27.165 ± 50.060	3.681 ± 5.977	175.609 ± 50.748	0.083 ± 0.106
SafeRL-TTFF-750	100.957 ± 95.943	1138.760 ± 411.799	6.224 ± 16.408	3.238 ± 5.863	61.450 ± 22.477	0.022 ± 0.054

Table 7. Mean QoE per route category (40 traces).

Route	RB	BBA	MPC-3	SafeRL-TTFF-750
Bus	156.26	109.26	107.88	143.86
Car	137.58	158.67	135.18	162.11

Ferry	83.89	-71.58	-109.36	55.18
Metro	55.52	-15.04	-154.57	47.28
Train	127.01	124.88	107.68	131.08
Tram 1	67.11	73.28	63.03	78.61
Tram 2	39.65	53.18	29.17	45.46
Tram 3	110.36	77.94	95.34	114.49

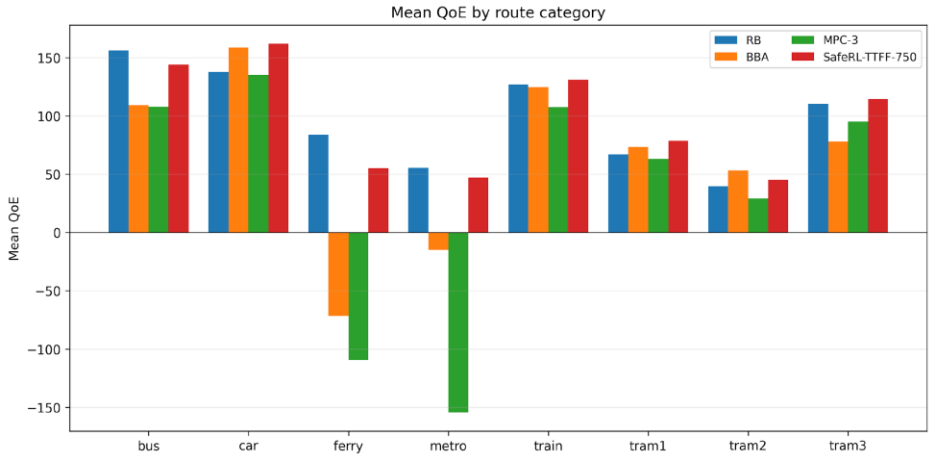

Fig. 5. Mean QoE by route category and controller.

Table 8 isolates startup and stall indicators on the held-out split. SafeRL-TTFF-750 reduces both mean and tail TTFF relative to RB: the mean drops from 2.221 s to 1.852 s and the 95th percentile from 2.609 s to 2.224 s. Its mean stall ratio remains close to zero (0.001), although RB still posts the smallest absolute rebuffering. The central practical tradeoff of the controller family is therefore clear: modestly higher stall risk than RB in exchange for shorter startup and slightly higher delivered quality.

Table 9 and Figure 6 clarify the role of the deployment layers. Removing the safety supervisor is catastrophic: mean rebuffering rises to 136.766 s and mean QoE becomes strongly negative. Safety is therefore not an auxiliary convenience; it is the mechanism that makes the learned policy usable on these traces.

Once the safety supervisor is present, the startup cap becomes a tunable service-level knob rather than a stability requirement. Safety-only SafeRL attains the highest scalar QoE among the ablation variants (136.647), but it starts more slowly. Adding a 750 kbps startup cap lowers TTFF substantially at only a small QoE cost. A stricter 300 kbps startup cap lowers TTFF further and slightly improves scalar QoE relative to the 750 kbps cap. For that reason, the revised paper no longer treats the 750 kbps cap as a unique optimum; instead, it is described as the main operating point used in the four-way comparison because it avoids forcing the first rendered segment to the minimum representation. That preference for startup visual quality is not modeled separately in the scalar QoE objective.

The combined evidence suggests three design lessons. First, learned bitrate selection can be useful, but only when wrapped in transparent safeguards. Second, TTFF should be treated as an explicit operating-point choice rather than a hidden side effect of the steady-state controller. Third, scalar QoE averages must be interpreted carefully on small test splits: with only eight held-out traces and large standard deviations, the results are directional rather than definitive. The strongest claim supported by the data is not that SafeRL-TTFF universally dominates all conventional controllers, but that safety-constrained RL can reach attractive startup-quality-stall tradeoffs that are difficult to express with a single hand-written rule.

Table 8. TTFF percentiles and stall-related metrics on the held-out test set.

Method	**TTFF mean (s)**	**TTFF p50 (s)**	**TTFF p90 (s)**	**TTFF p95 (s)**	**SRR mean**	**Mean stall events**	**Mean rebufferin g (s)**
RB	2.221	2.176	2.519	2.609	0.0	0.125	0.025
BBA	0.933	0.916	1.175	1.177	0.011	0.25	2.77

MPC-3	2.534	2.255	3.155	3.422	0.045	8.625	11.512
SafeRL-TTFF-750	1.852	1.898	2.219	2.224	0.001	0.5	0.228

Table 9. Deployment-layer ablation on the held-out test set.

Variant	**Mean QoE**	**Mean avg bitrate (kbps)**	**Mean rebuffering (s)**	**Mean TTFF (s)**	**Mean smoothness (kbps)**
RL (no safety)	-393.846	1691.406	136.766	3.302	59.401
SafeRL	136.647	1204.74	0.14	2.506	50.998
SafeRL-TTFF-750	136.125	1201.771	0.228	1.852	51.943
SafeRL-TTFF-300	136.505	1195.469	0.03	0.933	53.414

SafeRL-TTFF-300 and SafeRL-TTFF-750 denote startup caps at 300 and 750 kbps, respectively, applied only until playback starts.

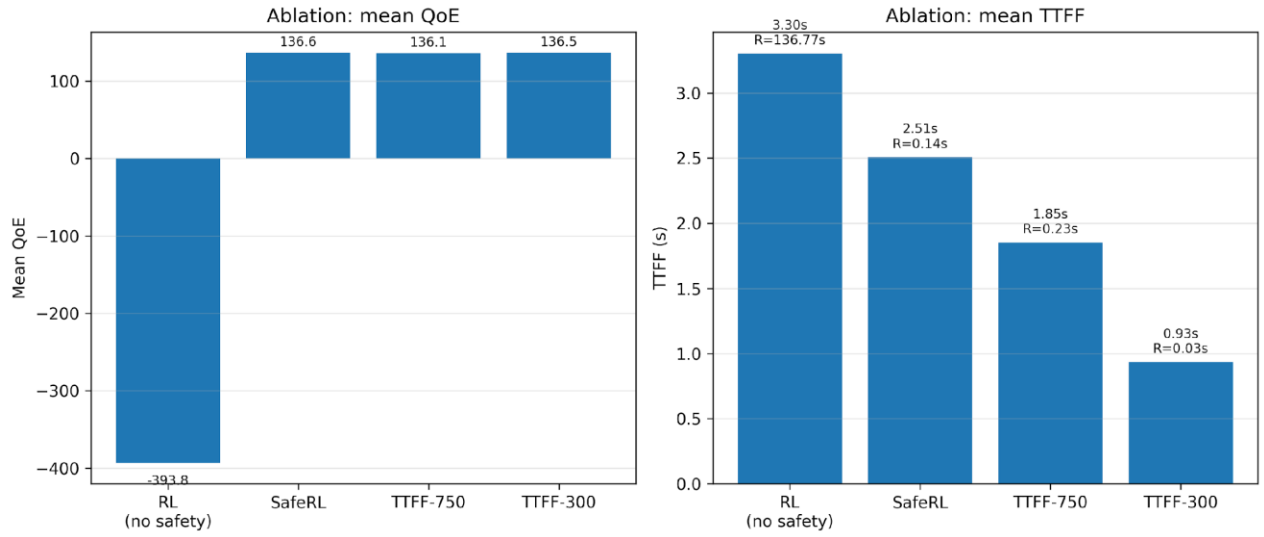

Fig. 6. Deployment-layer ablation on the held-out test set. Left: mean QoE. Right: mean TTFF, annotated with mean rebuffering.

4. Limitations

First, the evaluation is trace-driven and restricted to 40 selected traces from the MMSys'13 archive; it does not exhaust the full archive and does not represent newer cellular technologies. Second, the simulator uses fixed-size CBR segments, a fixed request overhead, and a simple startup threshold, whereas production players encounter variable-size segments, CDN latency variation, connection reuse effects, and in some cases partial-segment startup.

Third, both training and evaluation occur in the same simulator. Although the train/test split protects against direct memorization of the held-out traces, simulator-specific assumptions can still affect the absolute rankings. Fourth, the held-out set contains only eight traces and the paper does not report formal significance tests, so numerical rank differences should be interpreted cautiously.

Fifth, the scalar QoE model rewards bitrate and penalizes TTFF, rebuffering, and switching, but it does not separately value the visual quality of the first rendered segment. That omission matters when

comparing startup caps: a stricter cap may improve the scalar score yet still be less desirable for services that care about initial picture quality. Finally, the study does not address multi-client fairness, radio-state interactions, or implementation issues such as on-device model-update cost.

5. Conclusion

This paper revisited QoE-driven ABR on mobile HSDPA traces and studied a controller family that combines PPO-trained bitrate selection with an online safety supervisor and optional startup caps. The main four-way comparison shows that SafeRL-TTFF-750 improves held-out test-split QoE and TTFF relative to a throughput-based RB baseline while maintaining very low rebuffering.

The full-set comparison is more conservative: SafeRL-TTFF-750 and RB are essentially tied in mean QoE, with the learned controller shifting the operating point toward higher bitrate and shorter startup at the cost of more rebuffering. The ablation study is the clearest result. Safety supervision is indispensable, and startup caps provide a practical mechanism for moving along the TTFF-versus-initial-quality tradeoff.

Taken together, the revised evidence supports a restrained conclusion: reinforcement learning is promising for ABR not as a standalone replacement for classic logic, but as a policy component inside a safety-constrained and deployment-tunable controller.

References

[1] UMass Trace Repository, "HSDPA-bandwidth logs for mobile HTTP streaming scenarios (MMSys'13)," 2013. [Online]. Available: https://skulddata.cs.umass.edu/traces/mmsys/2013/pathbandwidth/

[2] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications," in Proc. ACM Multimedia Systems Conf. (MMSys), Oslo, Norway, 2013, pp. 114-118, doi: 10.1145/2483977.2483991.

[3] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning," ACM Trans. Multimedia Comput. Commun. Appl., vol. 8, no. 3, art. 24, 2012, doi: 10.1145/2240136.2240137.

[4] R. Pantos, W. May, and D. King, "HTTP Live Streaming," RFC 8216, Aug. 2017.

[5] ISO/IEC 23009-1:2022, "Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats," 2022.

[6] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," IEEE MultiMedia, vol. 18, no. 4, pp. 62-67, Oct.-Dec. 2011, doi: 10.1109/MMUL.2011.71.

[7] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," IEEE Commun. Surveys Tuts., vol. 17, no. 1, pp. 469-492, 2015, doi: 10.1109/COMST.2014.2360940.

[8] M.-N. Garcia, F. De Simone, S. Tavakoli, N. Staelens, S. Egger, K. Brunnström, and A. Raake, "Quality of Experience and HTTP Adaptive Streaming: A Review of Subjective Studies," in Proc. IEEE Int. Workshop on Quality of Multimedia Experience (QoMEX), Singapore, 2014, pp. 141-146, doi: 10.1109/QoMEX.2014.6982310.

[9] F. Dobrian, V. Sekar, A. Awan, I. Stoica, D. Joseph, A. Ganjam, J. Zhan, and H. Zhang, "Understanding the Impact of Video Quality on User Engagement," in Proc. ACM SIGCOMM, Toronto, ON, Canada, 2011, pp. 362-373, doi: 10.1145/2018436.2018478.

[10] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A Buffer-Based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in Proc. ACM SIGCOMM, Chicago, IL, USA, 2014, pp. 187-198, doi: 10.1145/2619239.2626296.

[11] J. Jiang, V. Sekar, and H. Zhang, "Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming with FESTIVE," in Proc. ACM CoNEXT, Nice, France, 2012, pp. 97-108, doi: 10.1145/2413176.2413189.

[12] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale," IEEE J. Sel. Areas Commun., vol. 32, no. 4, pp. 719-733, Apr. 2014, doi: 10.1109/JSAC.2014.140405.

[13] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in Proc. ACM SIGCOMM, London, UK, 2015, pp. 325-338, doi: 10.1145/2785956.2787486.

[14] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman, "BOLA: Near-Optimal Bitrate Adaptation for Online Videos," in Proc. IEEE INFOCOM, San Francisco, CA, USA, 2016, pp. 1-9, doi: 10.1109/INFOCOM.2016.7524428.

[15] H. Mao, R. Netravali, and M. Alizadeh, "Neural Adaptive Video Streaming with Pensieve," in Proc.

ACM SIGCOMM, Los Angeles, CA, USA, 2017, pp. 197-210, doi: 10.1145/3098822.3098843.

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," arXiv:1707.06347, 2017.

[17] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-Dimensional Continuous Control Using Generalized Advantage Estimation," in Proc. Int. Conf. Learning Representations (ICLR), 2016.

[18] V. Mnih et al., "Human-Level Control through Deep Reinforcement Learning," Nature, vol. 518, no. 7540, pp. 529-533, 2015, doi: 10.1038/nature14236.

[19] ITU-T, "Recommendation P.1203: Parametric Bitstream-Based Quality Assessment of Progressive Download and Adaptive Audiovisual Streaming Services over Reliable Transport," 2017.

[20] K. Spiteri, R. K. Sitaraman, and D. Sparacio, "From Theory to Practice: Improving Bitrate Adaptation in the DASH Reference Player," in Proc. ACM Multimedia Systems Conf. (MMSys), Amsterdam, Netherlands, 2018, doi: 10.1145/3204949.3204953.

[21] N. Bouten, J. Famaey, S. Latré, and F. De Turck, "QoE-Driven In-Network Optimization for Adaptive Video Streaming," Comput. Netw., vol. 81, pp. 284-297, 2015, doi: 10.1016/j.comnet.2015.02.012.

[22] A. Ahmed, Z. Shafiq, A. Srivastava, and A. Khakpour, "Suffering from Buffering? Detecting QoE Impairments in Live Video Streams," in Proc. IEEE ICNP, Toronto, ON, Canada, 2017, pp. 1-10, doi: 10.1109/ICNP.2017.8117575.

[23] Jing Chen, Xinzhuo Sun, and Vincent Brown, “Claim-Aware Scientific RAG: Evidence-First Retrieval and Abstention for Scientific Fact Responses on SciFact”, JACS, vol. 3, no. 1, pp. 16–30, Jan. 2023, doi: 10.69987/JACS.2023.30102.

[24] Binghua Zhou, Siming Zhao, and David Chao, “LLM-Guided Energy-Aware A/B Testing for Consolidation and DVFS Policies via Power-Sensitivity Clustering”, JACS, vol. 3, no. 4, pp. 12–30, Apr. 2023, doi: 10.69987/JACS.2023.30402.

[25] Daren Zheng, Chenyu Li, and Harvey Davidson, “Continual Red-Teaming for In-the-Wild Jailbreaks via Online Guardrail Updates and Guardrail Distillation”, JACS, vol. 3, no. 2, pp. 35–49, Feb. 2023, doi: 10.69987/JACS.2023.30203.

[26] Siming Zhao, Hailin Zhou, and Daniel Martinez, “LLM-Assisted Causal Attribution of Service Performance Upgrades on Churn and Tenure: Full Evaluation on the IBM Telco Customer Churn Dataset”, JACS, vol. 3, no. 2, pp. 18–34, Feb. 2023, doi: 10.69987/JACS.2023.30202.

Appendix A. Selected trace filenames

The experiments use the following 40 traces from the MMSys’13 archive. Listing the filenames explicitly ensures that the reported route-level summary can be reproduced exactly from the trace subset analyzed in this paper.

Bus (7): report.2010-09-28_1407CEST.log; report.2010-09-29_0852CEST.log; report.2010-09-29_1622CEST.log; report.2010-09-29_1628CEST.log; report.2010-09-29_1823CEST.log; report.2010-09-29_1827CEST.log; report.2010-09-30_1058CEST.log.

Metro (7): report.2010-09-13_1003CEST.log; report.2010-09-13_1046CEST.log; report.2010-09-14_1038CEST.log; report.2010-09-14_1415CEST.log; report.2010-09-14_2303CEST.log; report.2010-09-21_0742CEST.log; report.2010-09-22_0857CEST.log.

Tram ljabru-jernbanetorget (5): report.2010-11-23_1541CET.log; report.2010-12-09_1244CET.log; report.2010-12-09_1334CET.log; report.2010-12-16_1125CET.log; report.2010-12-16_1215CET.log.

Tram jernbanetorget-ljabru (4): report.2010-11-23_1515CET.log; report.2010-12-09_1222CET.log; report.2010-12-09_1310CET.log; report.2010-12-16_1100CET.log.

Tram jernbanetorget-universitetssykehuset (1): report.2010-11-23_1606CET.log.

Ferry (4): report.2010-09-20_1542CEST.log; report.2010-09-21_1001CEST.log; report.2010-09-21_1622CEST.log; report.2010-09-21_1735CEST.log.

Car (7): report.2011-02-10_1611CET.log; report.2011-04-21_1135CEST.log; report.2011-02-14_2032CET.log; report.2011-02-14_2051CET.log; report.2011-02-14_2108CET.log; report.2011-02-14_2124CET.log; report.2011-02-14_2139CET.log.

Train (5): report.2011-02-11_1618CET.log; report.2011-02-14_0644CET.log; report.2011-02-11_1530CET.log; report.2011-02-11_1729CET.log; report.2011-02-14_1728CET.log.