Into the Wild: Real-World Testing for ML-Based ABR

Benjamin Hoffman
ETH Zürich
bhoffman@ethz.ch

Alexander Dietmüller
ETH Zürich
adietmue@ethz.ch

Ayush Mishra
ETH Zürich
aymishra@ethz.ch

Laurent Vanbever
ETH Zürich
lvanbever@ethz.ch

Abstract

Machine learning (ML)-based Adaptive Bitrate (ABR) algorithms often struggle to bridge the gap between simulation and reality. Their strong performance in simulated or emulated environments frequently fails to generalize to real-world network conditions. Researchers have therefore begun testing these algorithms over the Internet to incorporate real-world feedback into their design. In this paper, we show that since network conditions vary significantly across the globe, testing in individual real-world environments can suffer from the same generalization issues as lab-based testing. Existing testing platforms suffer from (and might even be oblivious to) this limitation because they cover a small geographical region and rely on a narrow set of users affected by survivorship bias. As a result, their insights on an algorithm's performance generalize poorly to deployments in other environments across the Internet, hindering the widespread adoption of ML-based ABR methods in practice.

To address this gap, we present ABR-Arena, a global testing platform that enables researchers to evaluate the performance of ABR algorithms across a diverse set of regions around the globe. As a result of its worldwide coverage, ABR-Arena can reveal the performance shortcomings of several state-of-the-art ML-based approaches. It is extensible and easy to deploy in additional locations. We will make ABR-Arena available to the community to support the development of new ML-based approaches and to facilitate meaningful improvements to existing algorithms.

1 Introduction

Video streaming is the most prominent workload on the Internet, accounting for over 65% of downstream traffic [10]. Consistently providing high Quality-of-Experience (QoE) has therefore become critical for content providers seeking to maintain user engagement [4]. To maximize a user's QoE, Adaptive Bitrate (ABR) algorithms that dynamically adjust their sending behavior in response to changing network conditions are commonly used in practice. This typically involves selecting the appropriate bitrate to minimize delays and stall time at the client, while maximizing video quality.

PACMI'25, Seoul, Republic of Korea
2025. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

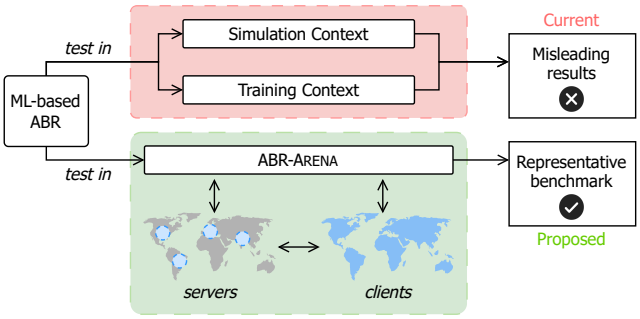

Figure 1. ABR-Arena enables efficiently evaluating algorithms in environments across the globe, allowing for more representative performance benchmarks.

Optimizing video QoE over the Internet is a difficult problem for classical heuristics-based algorithms. They must contend with a high-dimensional parameter space that spans network conditions, user behavior, device capability, and video characteristics, making it hard to model accurately.

This complexity creates an opportunity for ML-based methods to shine. They can rapidly interpret vast amounts of data, process large modeling spaces, and replace heuristic tuning with learning from experience. On this premise, research has increasingly shifted away from classical methods towards using ML-based ABR algorithms [7–9, 12, 14, 18].

The need for real-world feedback. However, the adoption of ML-based ABR schemes is hampered by a lack of real-world feedback. While many algorithms have shown promise when tested within their original training environment, they often fail to perform when deployed in practice [3, 6, 12]. One possible approach could be to evaluate them in *representative* synthetic environments through simulation or emulation. However, in the context of networking, creating representative environments for testing or training is challenging: it has been shown that the complex and heavy-tailed nature of Internet traffic makes it particularly hard to replicate in a controlled setting [1, 14, 15]. In other words, to evaluate the true performance of an ML-based algorithm, we need to test it in real-world scenarios. Therefore, the current strategy for making these algorithms suitable for the Internet is to directly train and evaluate them *in situ*, on the Internet itself.

Projects like Puffer [14] currently lead the charge on this front. Puffer is an open-source YouTube-like service that streams live television to real users across the United States using both classical and ML-based ABR algorithms. Several state-of-the-art ML-based approaches, in particular Fugu [14], Maguro [8], and Unagi [8], have been shown to outperform classical buffer-based or throughput-based ABR schemes [5, 11, 17] when trained using Puffer data and tested within Puffer's real-world environment. Fugu's creators report consistently outperforming all tested classical methods[1] in terms of video quality, and all but one in terms of stall time [14]. Similarly, Maguro's creators even claim to achieve 78% lower stall time than Fugu, as well as improved video quality [8]. To this date, Maguro and Unagi remain the best-performing algorithms on the Puffer platform, showcasing the benefits of using ML-based ABR algorithms over the Internet.

Despite its merits, Puffer has three main limitations hindering its results from generalizing to deployments in practice.

1. **Lack of regional diversity.** The Puffer infrastructure is deployed on a single server in Stanford and its viewership is restricted to the United States, which limits the platform's ability to capture a representative set of global network conditions. As we will show in §3, this lack of diversity can be a major hurdle when measuring an ABR algorithm's performance in practice.
2. **Survivorship bias.** The Puffer data also suggests the presence of survivorship bias in their users. For example, the study's proposed algorithm Fugu improved by roughly 50% in terms of stall ratio when comparing February of 2025 to its initial performance in February of 2019 without any retraining. While the study initially attracted a diverse set of users at its launch, Fugu's performance improvement over time indicates that users experiencing higher QoE are more likely to continue using the platform, hence biasing the results.
3. **Hard to deploy.** Finally, since Puffer is designed to be highly available and production-grade, in the author's own words [16], it is *non-trivial* to build and deploy. It is therefore cumbersome to replicate from scratch in multiple regions to gather more diverse data.

These limitations can leave researchers oblivious to their algorithm's performance in real-world environments outside of Puffer. Moreover, since Puffer is often used as the de facto state-of-the-art data collection and testing platform for ABR algorithms, its limitations also risk misguiding the designs of future ML-based algorithms via performance feedback that might be too specific to its narrow context and coverage.

Our Approach. To address this gap in algorithm evaluation, as well as the shortcomings in prior work, we propose ABR-Arena, a Python-based testing infrastructure for the efficient evaluation and comparison of ABR algorithm performance across diverse real-world environments (fig. 1). By containerizing streaming servers and deploying them to cloud instances worldwide, we design ABR-Arena to be easy to use and to extend to new locations. We mitigate the impact of survivorship bias by not relying on returning users, but rather stream to random users sourced via Amazon Mechanical Turk (MTurk), a popular crowdsourcing marketplace.

In this paper, we use ABR-Arena to evaluate state-of-the-art ML-based ABR algorithms in four diverse regions. We demonstrate that algorithms trained on Puffer's data might generalize poorly to other real-world environments, in some cases even losing their advantage over classical schemes.

To summarize, we make the following key contributions:

1. We propose ABR-Arena, an infrastructure for efficiently evaluating multiple ABR algorithms across diverse environments around the globe.
2. We address the shortcomings of previous work by deploying our streaming servers in multiple continents and streaming to real users globally, ensuring coverage of a large diversity of network conditions – similarly to what Pantheon did for congestion control (CC) [15]. To avoid the presence of survivorship bias, we stream to QoE-insensitive users sourced via MTurk.
3. We demonstrate the effectiveness of our approach by evaluating the performance of three state-of-the-art ML-based ABR algorithms across four real-world environments in Europe, the Americas, and Asia.
4. We show that an algorithm's performance in a single environment – especially its training environment – can vary greatly from its results in other contexts, in some cases performing worse than non-ML methods.

We plan to make ABR-Arena available to researchers around the world to aid the development of ML-based ABR algorithms and their adoption in practice by providing more diverse real-world feedback on their performance.

2 System Design

We design ABR-Arena to address three key challenges (fig. 2): *(i)* providing a higher diversity of real-world testing environments, *(ii)* mitigating the presence of survivorship bias, and *(iii)* ensuring ease of deployment and extensibility to additional locations and ABR algorithms.

Backbone. ABR-Arena consists of four key components: a Python interface, a deployable streaming server, a deployment and monitoring pipeline, as well as a QoE data collection and evaluation pipeline (fig. 2). In our streaming server implementation, we extend the infrastructure made available by Puffer and use their pre-embedded video sources. To make testing via crowdsourced users straightforward, we extend streaming support to all major browsers and prevent browser-based background throttling. By randomly assigning an algorithm to each streaming session, we maintain

[1] Fugu outperforms BBA [5], MPC-HM and RobustMPC-HM [17] in terms of video quality and all but RobustMPC-HM in terms of stall time.

Puffer's randomized controlled trial property. For deployment, we containerize our streaming server and push it to Docker Hub. We build our deployment and monitoring, as well as our data collection and evaluation pipelines in Python, using the netUnicorn library and services [2]. This setup allows us to simultaneously deploy our servers on multiple cloud instances across the globe and monitor the experiments on a rolling basis, using rsync to continuously transfer QoE measurement data to our local machine for evaluation.

Regional diversity. We want to use ABR-Arena to test ABR algorithms in a variety of real-world environments. To this end, for the preliminary evaluations presented in this paper, we deployed our streaming servers to cloud instances in Sao Paulo, Zurich, Mumbai, and Ohio. We chose this mix as a good starting point that covers diverse geographical locations. However, thanks to ABR-Arena's extensibility, it can easily be expanded to more locations globally as well.

Crowdsourcing an unbiased user base. Unlike Puffer, which streams to a returning audience, we field random users via Amazon's MTurk service, a crowdsourcing marketplace that provides access to a broad set of Internet users to complete virtual tasks. As our users stem from a paid platform, we mitigate the presence of survivorship bias in our results. In other words, while results from Puffer indicate that users with a good streaming experience tend to return more frequently, narrowing the diversity of sessions and biasing the results, our users are insensitive to QoE and are not affected by the same mechanism. As this restricts us from capturing additional statistics on user behaviour, we focus primarily on measuring QoE performance metrics. By using MTurk, we can also handpick the geographical location of our user base for each experiment. Further, we ask users to provide additional information on how (wired, wifi, cellular) and where (residential, work, university) their device is connected to the Internet as metadata. In our preliminary results presented here, we fielded 11,156 users from 93 countries. Overall, our measurements cost us roughly 500 USD.

Ease of deployment and extensibility. We design ABR-Arena to be light-weight and easily deployable. This allows it to be extended to new locations globally and not fall into the same generalization pitfalls Puffer has over the years. A user can easily interact with the system by setting the desired configurations, e.g., which ABRs to test, their weights or variants, which CC algorithm to use, in which locations to deploy the servers, which cloud providers to use, how long to stream to each user, etc. Based on these inputs, ABR-Arena deploys the desired streaming servers, while the user can monitor the infrastructure and evaluate the results on their local machine in real time. New testing locations can be readily added to ABR-Arena via a template (often, a one-time setup on the cloud provider's platform is required), and we maintain Puffer's ability to host new ABR schemes.

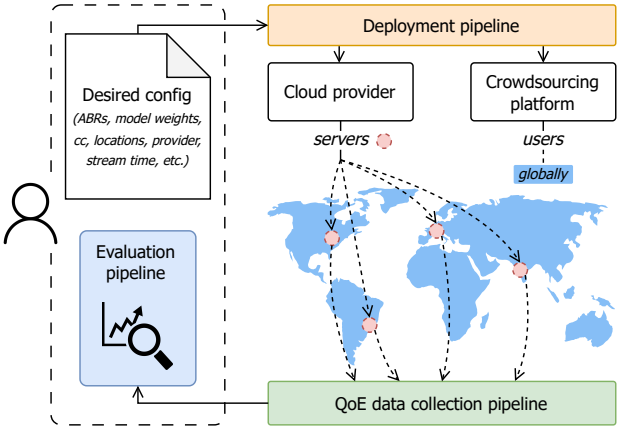

Figure 2. With ABR-Arena, we provide access to more diverse testing environments for ABR algorithms by enabling easy deployment to servers across the globe, as well as maintaining extensibility to further locations and ABR schemes.

3 Preliminary Results

We demonstrate the effectiveness of ABR-Arena by evaluating the performance of Fugu,[2] Maguro, and Unagi across four real-world environments. Fugu combines ML-based throughput prediction with an MPC controller to form a hybrid ABR approach and was trained on Puffer data. Maguro and Unagi are two RL-based algorithms. Both are similarly trained in simulation using Puffer traces, Unagi using randomly chosen traces, and Maguro using sampling intended to address dataset skewness. As a non-ML baseline, we deploy BBA, a simpler buffer-based ABR algorithm [5]. We use ABR-Arena to deploy our streaming infrastructure to AWS instances in Zurich, Ohio, Sao Paulo, and Mumbai, and collect QoE measurements by streaming to users across the globe. Additionally, we compare our results to the performance these algorithms achieve on Puffer. In our experiments, we collect 510 streaming hours across 11,156 users, each streaming for 2 minutes and 45 seconds on average, between February and June of 2025. Given the uneven distribution of streaming hours (302 hours in Zurich, 95 hours in Ohio, 93 hours in Sao Paulo, 20 hours in Mumbai), we compute metrics per streaming session and compare their averages per environment.

Performance metrics. To compare the performance of an ABR algorithm, we evaluate the QoE it provides via two major metrics shown to drive a user's engagement with video content: the video quality and the buffering time [4, 11]. We use SSIM[3] *(higher is better)* to measure the perceived quality of a video [13], and the stall ratio *(lower is better)*, i.e., the percentage of time spent stalling, to measure buffering time.

[2]We use Fugu_{feb}, a Fugu variant trained on Puffer data from February 2019 and used in the experiments in the original paper [14].
[3]As in [14], we convert the standard SSIM metric to a decibel scale.

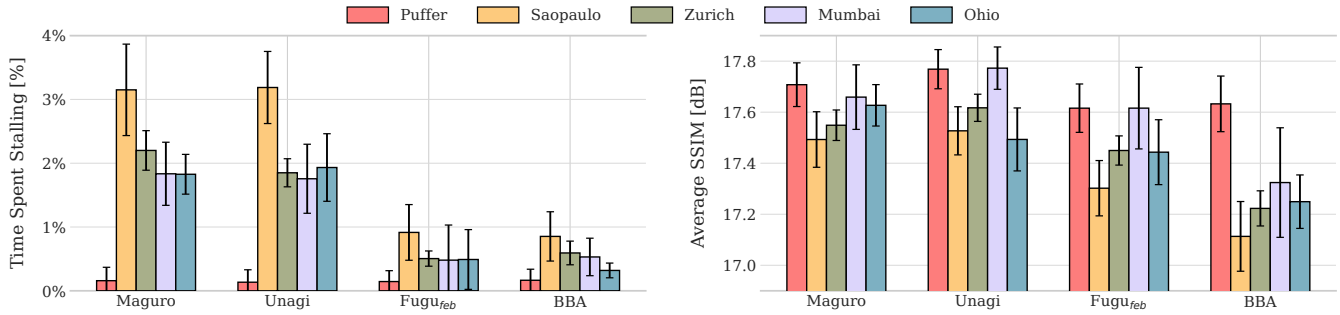

Figure 3. The large variations in stall ratio *(left – lower is better)* and video quality *(right – higher is better)* across deployments show that testing in a single environment does not allow for a representative evaluation of an ABR's performance. In particular, an algorithm's performance in its training context — here Puffer — can be markedly different from its performance in practice. We plot the mean and the bootstrapped 95% confidence intervals for each algorithm in each environment.

Variance across environments. The QoE performance of all algorithms varies markedly across our environments, both absolutely and relatively (fig. 3). This is especially pronounced for the RL-based schemes, Maguro and Unagi, with their variance in stall ratio across our environments being roughly 10 times higher than Fugu's. While the Sao Paulo environment proves the most challenging, regional proficiency varies between algorithms. For instance, Maguro performs worse in Ohio than Zurich, while Unagi shows the opposite trend. Fugu, thanks to its hybrid design, is more robust to environmental changes, but still not immune to them. In Sao Paulo, Fugu's stall ratio is 86.4% higher and its SSIM 0.14 dB lower than in Ohio. Similarly, our non-ML baseline, BBA, varies less than Maguro and Unagi in terms of stall ratio.

Training and deployment gap. To assess how algorithms generalize beyond their training contexts, we compare their performance in ABR-Arena against their performance on Puffer. To this end, we analyzed 23,236 streaming hours of Puffer data from March 2025. Across the board, stall ratios are worse in our environments compared to Puffer (fig. 3). As before, this gap is especially stark for both RL-based algorithms, Maguro and Unagi, compared to Fugu, which is considerably more resilient. Although both Maguro and Unagi outperform Fugu on Puffer, in our experiments in ABR-Arena, their stall ratios are significantly higher than Fugu's – by 276.8% and 264.8%, respectively. The differences in SSIM between Puffer and ABR-Arena were smaller, but still present.

When considering BBA, the results are even more surprising. Despite coming last on Puffer, BBA's stall reduction outperforms both Maguro and Unagi in our environments, equalling Fugu. While BBA's SSIM is lower, it strikes a better trade-off between stalling and quality than Maguro and Unagi in challenging environments, whose stall ratios reach $> 3\%$ in Sao Paulo. Only Fugu offers similarly robust performance. These results suggest that, due to its limitations, testing on Puffer does not necessarily reveal the sensitivity of ML-based ABRs to Out-of-Distribution (OOD) environments.

4 Discussion

Overall, our results highlight both the value and necessity of ABR-Arena: QoE performance varies significantly across real-world environments, and comparisons with Puffer reveal how much ML-based ABR performance can diverge between training and deployment. A reliable evaluation of these algorithms can only be done by testing them across diverse regions with varying network conditions.

However, achieving *sufficient* diversity during evaluation remains a challenge. While ABR-Arena improves on previous work and, by its extensible design, allows it to grow more diverse in the future, it has its limitations. Our current deployments — limited to major cloud providers and a university network — may not reflect production environments of large streaming platforms. Our preliminary dataset is also smaller than the Puffer dataset, despite covering more regions. We plan to remedy this by continuing to capture more results to further substantiate our findings. Finally, while we measure the performance gap that can exist between training and deployment, with some algorithms even losing their edge over classical methods, we do not offer any solutions on how to close it. ABR-Arena can perhaps aid here as well, by helping collect more diverse *training* data from different environments to learn algorithms that *generalize* to deployments across the Internet. This remains future work.

5 Conclusion

In this work, we present ABR-Arena, a global testing platform for evaluating (ML-based) ABR algorithms across a diverse set of real-world environments. Using ABR-Arena, we address and reveal the variance of an algorithm's QoE performance between its training environment and its deployment, as well as across different geographical regions. By designing our platform to be easy to use, to deploy, and to extend, we hope to support researchers in testing, developing, and adopting new ML-based approaches that offer meaningful improvements over existing schemes.

References

[1] Mihovil Bartulovic, Junchen Jiang, Sivaraman Balakrishnan, Vyas Sekar, and Bruno Sinopoli. 2017. Biases in Data-Driven Networking, and What to Do About Them. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks* (Palo Alto, CA, USA) *(HotNets '17)*. Association for Computing Machinery, New York, NY, USA, 192–198. doi:10.1145/3152434.3152448

[2] Roman Beltiukov, Wenbo Guo, Arpit Gupta, and Walter Willinger. 2023. In Search of netUnicorn: A Data-Collection Platform to Develop Generalizable ML Models for Network Security Problems. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (Copenhagen, Denmark) *(CCS '23)*. Association for Computing Machinery, New York, NY, USA, 2217–2231. doi:10.1145/3576915.3623075

[3] Paul Crews and Hudson Ayers. 2018. CS244'18: Recreating and Extending Pensieve. https://reproducingnetworkresearch.wordpress.com/wp-content/uploads/2018/07/recreating_pensieve.pdf

[4] Florin Dobrian, Vyas Sekar, Asad Awan, Ion Stoica, Dilip Joseph, Aditya Ganjam, Jibin Zhan, and Hui Zhang. 2011. Understanding the impact of video quality on user engagement. In *Proceedings of the ACM SIGCOMM 2011 Conference* (Toronto, Ontario, Canada) *(SIGCOMM '11)*. Association for Computing Machinery, New York, NY, USA, 362–373. doi:10.1145/2018436.2018478

[5] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: evidence from a large video streaming service. In *Proceedings of the 2014 ACM Conference on SIGCOMM* (Chicago, Illinois, USA) *(SIGCOMM '14)*. Association for Computing Machinery, New York, NY, USA, 187–198. doi:10.1145/2619239.2626296

[6] Hongzi Mao, Shannon Chen, Drew Dimmery, Shaun Singh, Drew Blaisdell, Yuandong Tian, Mohammad Alizadeh, and Eytan Bakshy. 2019. Real-World Video Adaptation with Reinforcement Learning. In *Proceedings of the ICML 2019 Workshop on Reinforcement Learning for Real Life (RL4RealLife)*.

[7] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication* (Los Angeles, CA, USA) *(SIGCOMM '17)*. Association for Computing Machinery, New York, NY, USA, 197–210. doi:10.1145/3098822.3098843

[8] Sagar Patel, Junyang Zhang, Nina Narodystka, and Sangeetha Abdu Jyothi. 2024. Practically High Performant Neural Adaptive Video Streaming. *Proc. ACM Netw.* 2, CoNEXT4, Article 30 (Nov. 2024), 23 pages. doi:10.1145/3696401

[9] Felipe Rosa, Simone Ferlin, Anna Brunstrom, and Bruno Kimura. 2025. End-to-End 360° Video Streaming over HTTP/3: Architecture and Implementation. In *Proceedings of the 2025 Applied Networking Research Workshop* (Madrid, Spain) *(ANRW '25)*. Association for Computing Machinery, New York, NY, USA, 9–16. doi:10.1145/3744200.3744784

[10] Sandvine Corporation. 2023. Video Permeates, Streaming Dominates. 14–15 pages. https://www.sandvine.com/hubfs/Sandvine_Redesign_2019/Downloads/2023/reports/SandvineGIPR2023.pdf

[11] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2020. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. *IEEE/ACM Transactions on Networking* 28, 4 (2020), 1698–1711. doi:10.1109/TNET.2020.2996964

[12] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving Video Bitrate Selection and Adaptation with Data-Driven Throughput Prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference* (Florianopolis, Brazil) *(SIGCOMM '16)*. Association for Computing Machinery, New York, NY, USA, 272–285. doi:10.1145/2934872.2934898

[13] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing* 13, 4 (2004), 600–612. doi:10.1109/TIP.2003.819861

[14] Francis Y. Yan, Hudson Ayers, Chenzhi Zhu, Sadjad Fouladi, James Hong, Keyi Zhang, Philip Levis, and Keith Winstein. 2020. Learning in situ: a randomized experiment in video streaming. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*. USENIX Association, Santa Clara, CA, 495–511. https://www.usenix.org/conference/nsdi20/presentation/yan

[15] Francis Y. Yan, Jestin Ma, Greg D. Hill, Deepti Raghavan, Riad S. Wahby, Philip Levis, and Keith Winstein. 2018. Pantheon: the training ground for Internet congestion-control research. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. USENIX Association, Boston, MA, 731–743. https://www.usenix.org/conference/atc18/presentation/yan-francis

[16] Francis Y. Yan and the Stanford Network Research Group. 2020. Puffer Documentation. https://github.com/StanfordSNR/puffer/wiki/Documentation Accessed July 29, 2025.

[17] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication* (London, United Kingdom) *(SIGCOMM '15)*. Association for Computing Machinery, New York, NY, USA, 325–338. doi:10.1145/2785956.2787486

[18] Hiba Yousef, Jean Le Feuvre, and Alexandre Storelli. 2020. ABR prediction using supervised learning algorithms. In *2020 IEEE 22nd International Workshop on Multimedia Signal Processing (MMSP)*. 1–6. doi:10.1109/MMSP48831.2020.9287123