# BURST OUT OF THE BEHAVIOR: AN ADAPTING SOLUTION FOR ON-DEMAND STREAMING

*Eduard Anton* and *Shu Tian*

McGill University, Montreal, QC, Canada

## ABSTRACT

Exploring the challenges associated with bursty behavior in HTTP Adaptive Streaming (HAS) for on-demand video content, the spotlight is on the impact of traditional adaptive streaming algorithms. In the context of video streaming dominating network traffic, the bursty nature of these algorithms can lead to congestion and disruptions. Solutions, such as application-informed pacing and adaptive bitrate algorithms, are investigated for their potential to dynamically adjust the server's sending rate and optimize bitrates based on network conditions. The primary goal is to enhance the Quality of Experience (QoE) while addressing the network efficiency challenges inherent in HAS for on-demand streaming.

***Index Terms***— Adaptive streaming, video streaming, TCP, adaptive bitrate, application-informed pacing, quality of experience, video on-demand

## 1. INTRODUCTION

Since 2022, video streaming has accounted for over 80% of all networking traffic, as reported by Cisco [1]. In the scenario where every user opted for the highest video quality then the resulting network traffic would be overwhelming, leading to escalated network investment costs and usage fees for streaming providers. This surge in demand has prompted the development of numerous innovative solutions geared towards enhancing the user experience on large streaming platforms such as Netflix, Youtube and Hulu. Recent solutions have predominantly focused on achieving a delicate equilibrium between traffic volume and video quality to ensure sustainable and cost-effective streaming services. Specifically, on-demand video streaming technologies, the focus for this paper, use adaptive streaming methods such as dynamic bitrate and pacing depending on the the measured performance of the user's video player. These adaptive approaches empower the system to dynamically tailor the video quality to match the available network conditions while limiting the usage of network resources to increase the overall performance of the common network for all users. This paper explores the latest on-demand streaming technologies and their influence on both Quality of Experience (QoE) and network performance.

## 2. METRICS AND PARAMETERS

### 2.1. Quality of Experience

The Quality of Experience (QoE) in video streaming is a crucial metric that encapsulates the user's overall satisfaction while consuming media content on platforms like YouTube or Netflix. Users anticipate a seamless and enjoyable experience, which is quantifiable through factors such as video quality, start-play delay, and rebuffers [2]. Video quality pertains to the format and clarity of the video which influences the visual experience. Start-play delay measures the time it takes for the video to load and start playing, while rebuffers quantify the number of interruptions the users face during video playback. Ideally, optimizing QoE involves maximizing video quality and minimizing start-play delay and rebuffers. However, achieving an optimal balance is challenging, as improvements in one aspect may negatively affect others [2]. For instance, reducing rebuffers and enhancing video quality often extends start-play delay, as a larger video buffer is required for optimization. Similar trade-offs apply when selecting different aspects of the QoE, with the realization that no more than two aspects can be optimized simultaneously due to inherent limitations.

### 2.2. Bitrate, Chunks, and Buffers

When users engage with video content on a platform, a buffer on their device stores individual chunks received from a server. These chunks, typically of specific duration, constitute blocks of media essential for smooth playback. Each of these blocks carries a bitrate, representing the amount of information needed to process the video chunk, usually measured in bits per second. The bitrate serves as a crucial determinant of the quality of the media within the chunk, with higher bitrates translating to improved media quality [2]. However, it's important to note that an increase in bitrate not only enhances media quality but also increases the data size of the chunk. As seen in Table 1, the bitrate usually increases with the image resolution, but it is not defined solely by it.

| Resolution | Min. Bitrate | Max. Bitrate |
|---|---|---|
| 2160p | 8 | 35 |
| 1440p | 5 | 25 |
| 1080p | 3 | 8 |
| 720p | 3 | 8 |

**Table 1**. Bitrate (Mbps) for different resolutions at 30fps on YouTube [4]

## 3. HTTP ADAPTIVE STREAMING

HTTP Adaptive Streaming (HAS) has emerged as a key solution for delivering high-quality video content over the internet, particularly in the context of Video on Demand (VoD). Unlike live streaming, VoD allows users at any time to choose and consume content that was already produced and published on a content delivery platform. Various transport layer technologies can be used to deliver VoD. On one hand, there is UDP-based streaming which offers low latency but also suffers from reliability issues due to its susceptibility to packet losses, leading to reduced visual fidelity [3]. In particular, Real-time Transport Protocols (RTP) often utilize UDP for real-time streaming, prioritizing immediacy over robustness [1]. On the other hand, there is TCP-based streaming which introduces reliability to UDP, however it is provided at the cost of speed since TCP introduces an acknowledgment process which prioritizes the integrity of data transmission over rapid delivery. Regardless, research indicates that when the available TCP throughput is double the media bitrate, the adverse effects of TCP impairments can be mitigated [3]. However, blindly maximizing network throughput can lead to congestion, affecting the fairness and performance of other clients and processes [2]. Existing literature suggests two primary approaches to maintaining optimal throughput and QoE: enhancing video compression efficiency [3] and implementing adjustable bitrates. This paper primarily delves into the latter solution, exploring how adaptive bitrate mechanisms contribute to an improved QoE by dynamically adjusting video quality based on available network conditions.

As first introduced in section 2.2, HAS techniques operate by breaking down a video file into a series of manageable chunks, with each chunk encoded at various bitrates. As presented in Fig. 1, when a user requests a video, the server responds with a Media Presentation Description (MPD), typically in XML format, containing the details of the upcoming video chunks [3]. The client then begins the streaming process by sequentially receiving in-order and storing these chunks into a buffer for playback. In order to reduce the latency time, the streaming service might also employ a content delivery network (CDN) proxy to cache clips [3]. During initialization, otherwise known as the buffering state, the client's buffer starts empty and quickly stars getting filled up until reaching a minimum threshold before playback commences [3]. This buffering phase ensures a smooth streaming

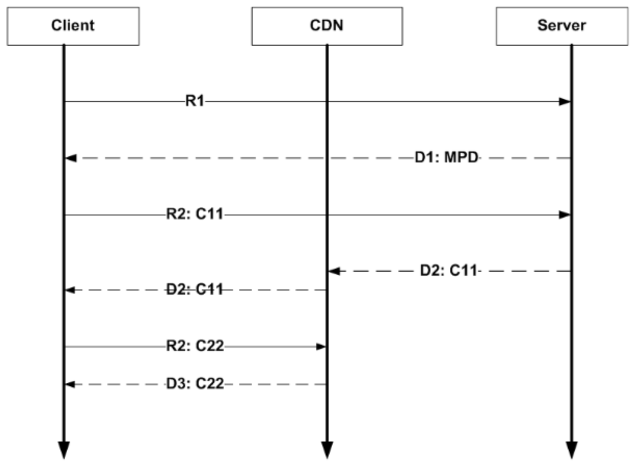

**Fig. 1**. Typical sequence diagram of HAS during streaming session [3]

experience by allowing a sufficient amount of video data to be pre-loaded, minimizing the risk of interruptions due to network fluctuations. Once the initial buffer is filled, the client continues into the steady-state to receive and store subsequent chunks, ensuring a continuous and uninterrupted playback experience for the user.

All HAS implementations behave differently depending how they balance QoE aspects. For example, under the same conditions and variations, Microsoft's Smooth Streaming starts playing 3 seconds in and fills up its buffer after 40 seconds during buffering [5]. Meanwhile, Netflix's playback starts playing almost 13 seconds in and never ends its buffering state for the first 500+ seconds [5], but its bitrate selection profile is more assertive, implying Netflix's focus in delivering higher quality video.

## 4. BURSTY BEHAVIOR

Typically, HAS algorithms operate at the application layer and follow a progressive dispatch approach which involves continuously receiving the next video chunks at the maximum throughput permitted by the network. Historically, to obtain optimal QoE, this design choice was encouraged because of the known performance limitations and network instabilities of older technologies, but now it is deemed unfavorable given the improved performance of modern systems [3]. In this scenario, as showcased in Fig. 2, the client's buffer rapidly fills up and is no longer able to store further chunks until playback catches up, resulting in an on-and-off behavior, commonly referred to as bursty behavior. As the client's throughput increases, the duration of the on-periods (buffer filling) become shorter, while the duration of the off-periods (waiting time) extend. This approach introduces issues such as packet loss, unfairness among TCP clients, and significant queuing

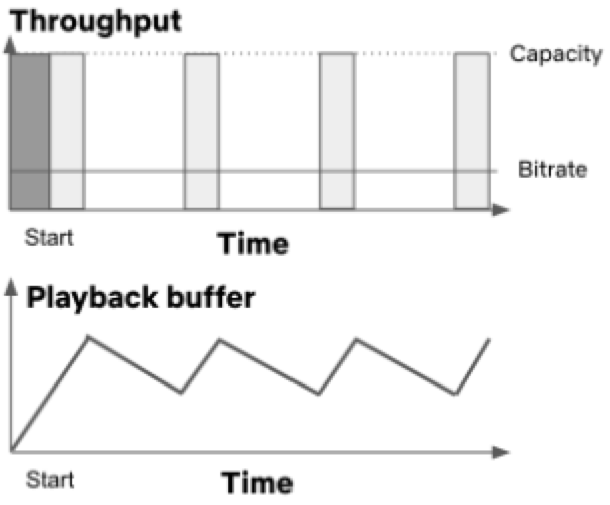

**Fig. 2**. Bursty behavior during typical streaming session [2]

delays, causing congestion. While the user in this scenario might be able to obtain acceptable QoE, the rest of the traffic on the network is left to suffer [2]. To address this, disposing of burstiness can be accomplished by maintaining throughput under the buffer's capacity but just above the current bitrate, to avoid rebuffers. Numerous studies have delved into devising optimal adaptive bitrate algorithms and application-informed pacing methods to adjust this delicate balance [2]. For instance, the proposed implementation for Sammy [2], an adaptation combining the two previously mentioned methods, was able to lower chunk throughput by 61% at the median, improving retransmissions and RTTs by 35% and by 14% respectively, while the overall QoE being slightly improved. The improved efficiency of the throughput, also enhanced the neighboring throughput of UDP by 51%, TCP by 28%, and HTTP by 18% [2].

## 5. APPLICATION-INFORMED PACING

While a straightforward approach of raising the bitrate can enhance QoE and decrease burstiness of a video streaming, it concurrently elevates network traffic, posing challenges for network performance [1]. In response, application-informed pacing emerges as a strategic solution, allowing applications to define an upper limit on the server's sending rate, directly impacting the throughput. Allowing to define the buffer size as $B_0$, the throughput as $x$, the lookahead duration of the upcoming $T$ chunks as $D_T$, and a parameter to offset prediction errors as $\beta \in [0, 1]$, at a given bitrate $r$, the client requires a minimum throughput defined in (1) to keep away rebuffers [2].

$$x \geq r\beta^{-1}\left(1 + \frac{B_0}{D_T}\right) \tag{1}$$

The HAS algorithm determines a pace rate which satisfies the minimum throughput and then forwards that information to the server through an HTTP header. This pace rate can be dynamically selected based on various metrics, such as buffer fill levels. An illustrative example is Sammy's utilization of TCP Pacing, achieved by configuring the SO_MAX_PACING_RATE socket option [2], dynamically adapting the sending pace based on the buffer's fill status. In particular, SO_MAX_PACING_RATE is a TCP policy provided by Linux and defines the pacing rate of TCP packets [6].

## 6. ADJUSTABLE BITRATE ALGORITHMS

The previous section introduced pacing to negate burstiness by essentially limiting the user's throughput. Unfortunately until now, its implementation had assumed that the bitrate in (1) was constant, but in reality, the bitrate is subject to continous change at any given time on a given network. In order to maintain the highest QoE for the users, the algorithm must be able to adapt to its environment by picking the highest bitrate available. As such, a combination of both application-informed pacing and adjustable birate algorithms would be sufficient to solve the burstiness behavior while maintaining an acceptable QoE.

As previously stated, a simple ABR system consists of a streaming server storing multiple profiles of the same video data encoded in different bitrates and quality levels, which are also divided into fragments of typically a few seconds long, and a client requesting the data [5]. When a video is played, the ABR algorithm selects a suitable bitrate for each chunk of data for the client to download sequentially. The appropriate bitrate may be selected based on various different factors at different times, depending on the algorithm [1].

### 6.1. Throughput Based

Throughput-based ABR algorithms are fundamental and widely used in the realm of adaptive streaming. Most prominent streaming platforms usually incorporate variants of throughput-based ABR into their service. Some algorithms like Sammy only need to handle approximate measurements of the client throughput [2], but most traditional approaches cannot and reveal the inherent difficulty in achieving accurate client-side throughput estimation above the HTTP layer, especially when dealing with wireless networks. Throughput-based ABR algorithms set the bitrate proportional to the current throughput. Both Sammy [2] and PANDA [3] have ABR logic that make them throughtput-based. The latter algorithm emulates TCP's congestion control mechanisms which operates at the application layer, as opposed to TCP which operates at the transport layer. In this way, it can obtain reliable measurements of the throughput and adjust the bitrate accordingly [3]. Other throughput-based ABR algorithms that exclusively deal with throughput heuristic include FESTIVE and QDASH-qoe [3].

### 6.2. Buffer Based

Buffer-based ABR is another pivotal approach in adaptive streaming, where the algorithm dynamically adjusts the bitrate based on the status of the client's buffer. This method focuses on dynamically adjusting the bitrate to optimize and maintain a consistently well-filled buffer, ensuring a smoother streaming experience with a low number of rebuffers [3]. Indeed, it has been shown that the probability of buffer getting emptied decreases exponentially with respect to the initial buffer level. Specifically, it was also demonstrated that introducing a buffer-based algorithm on Netflix allowed a decrease in rebuffers by 20% [3]. A few examples of buffer-based ABR include BBA0, BOLA, and Quetra [7].

### 6.3. Control Theory Based

Instead of dealing with heuristic or extremely hard to obtain measurements, some literature proposed instead an approach that involves control theory [3]. In short, control theory deals with modeling the behavior of dynamic systems. Using the difference between a defined measurement value and a target, an error can be obtained which can be used to adjust the bitrate of the stream. For example, the measurement could be the buffer level, in which case we would obtain a hybrid buffer and control theory based ABR algorithm [7].

### 6.4. Optimisation Based

Optimization-based ABR algorithms prioritize maximizing benefits and minimizing penalties in the bitrate adjustment process. Various metrics can guide the optimization, determining which values to enhance or minimize. The formalization of this behavior often involves linear integer programming [3]. A notable example is the NOVA algorithm, which falls within the optimization-based ABR category [3]. NOVA aims to maximize video quality while concurrently minimizing the variance in video quality [3].

### 6.5. Artificial Intelligence Based

AI-based ABR algorithms were introduced on the premise that traditional methods lack an inherent understanding of a user's viewing experience during video consumption. In contrast, AI ABR algorithms, employing deep learning (DL) and reinforcement learning (RL), present strategies aimed at directly enhancing Quality of Experience (QoE). One notable algorithm, ECAS-ML, stands out for its ability to predict adjustable parameter values, showcasing significant improvements over its naive counterparts as demonstrated in studies [8].

## 7. CONCLUSION

In summary, this report investigates the surge in video streaming, constituting over 80% of networking traffic since 2022 [1], and the challenges it poses to both QoE and network efficiency. In particular, the focus lies on dissecting the bursty behavior inherent in traditional adaptive streaming algorithms, where clients rapidly fill buffers, causing congestion and interruptions. While this approach may offer acceptable QoE for users, it adversely impacts the broader network. Fortunately, there exist many proposed solutions that would maintain the QoE but reduce the network traffic, including application-informed pacing, which dynamically adjusts the server's sending rate, and adaptive bitrate algorithms, based on throughput, buffer size, control theory logic, optimization or AI predictions.

## 8. REFERENCES

[1] T. Kimura, T. Kimura, A. Matsumoto, and K. Yamagishi, “Balancing quality of experience and traffic volume in adaptive bitrate streaming,” *IEEE Access*, vol. 9, pp. 15 530–15 547, 2021.

[2] B. Spang, S. Kunamalla, R. Teixeira, *et al.*, “Sammy: Smoothing video traffic to be a friendly internet neighbor,” in *Proceedings of the ACM SIGCOMM 2023 Conference*, ser. ACM SIGCOMM ’23, New York, NY, USA: Association for Computing Machinery, 2023, pp. 754–768, ISBN: 9798400702365. DOI: `10.1145/3603269.3604839`. [Online]. Available: `https://doi.org/10.1145/3603269.3604839`.

[3] Y. Sani, A. Mauthe, and C. Edwards, “Adaptive bitrate selection: A survey,” *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2985–3014, Fourthquarter 2017, ISSN: 1553-877X. DOI: `10.1109/COMST.2017.2725241`.

[4] [Online]. Available: `https://support.google.com/youtube/answer/2853702?hl=en`.

[5] S. Akhshabi, A. C. Begen, and C. Dovrolis, “An experimental evaluation of rate-adaptation algorithms in adaptive streaming over http,” in *Proceedings of the second annual ACM conference on Multimedia systems*, 2011, pp. 157–168.

[6] [Online]. Available: `https://man7.org/linux/man-pages/man8/tc-fq.8.html`.

[7] B. Taraghi, A. Bentaleb, C. Timmerer, R. Zimmermann, and H. Hellwagner, “Understanding quality of experience of heuristic-based http adaptive bitrate algorithms,” in *Proceedings of the 31st ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*, ser. NOSSDAV ’21, Istanbul, Turkey: Association for Computing Machinery, 2021, pp. 82–89, ISBN: 9781450384353. DOI: `10.1145/3458306.3458875`. [Online]. Available: `https://doi.org/10.1145/3458306.3458875`.

[8] J. A. Armijo, E. Çetinkaya, C. Timmerer, and H. Hellwagner, “ECAS-ML: edge computing assisted adaptation scheme with machine learning for HTTP adaptive streaming,” *CoRR*, vol. abs/2201.04488, 2022. arXiv: `2201.04488`. [Online]. Available: `https://arxiv.org/abs/2201.04488`.