Enabling adaptive bitrate algorithms in hybrid CDN/P2P networks

Hiba Yousef, Jean Le Feuvre, Paul-Louis Ageneau, Alexandre Storelli

Enabling adaptive bitrate algorithms in hybrid CDN/P2P networks

Hiba Yousef*†
Jean Le Feuvre*
*LTCI, Telecom Paris, Institut Polytechnique de Paris
Paris, France
firstname.lastname@telecom-paris.fr

Paul-Louis Ageneau†
Alexandre Storelli†
† Streamroot
Paris, France
firstname.lastname@streamroot.io

ABSTRACT

As video traffic becomes the dominant part of the global Internet traffic, keeping a good quality of experience (QoE) becomes more challenging. To improve QoE, HTTP adaptive streaming with various adaptive bitrate (ABR) algorithms has been massively deployed for video delivery. Based on their required input information, these algorithms can be classified, into buffer-based, throughput-based or hybrid buffer-throughput algorithms. Nowadays, due to their low cost and high scalability, peer-to-peer (P2P) networks have become an efficient alternative for video delivery over the Internet, and many attempts at merging HTTP adaptive streaming and P2P networks have surfaced. However, the impact of merging these two approaches is still not clear enough, and interestingly, the existing HTTP adaptive streaming algorithms lack testing in a P2P environment. In this paper, we address and analyze the main problems raised by the use of the existing HTTP adaptive streaming algorithms in the context of P2P networks. We propose two methodologies to make these algorithms more efficient in P2P networks regardless of the ABR algorithm used, one favoring overall QoE and one favoring P2P efficiency. Additionally, we propose two new metrics to quantify the P2P efficiency for ABR delivery over P2P.

CCS CONCEPTS

• **Information systems → Multimedia streaming**.

KEYWORDS

HTTP Adaptive Streaming, P2P, CDN, ABR, Response Delay, QoE

ACM Reference Format:
Hiba Yousef, Jean Le Feuvre, Paul-Louis Ageneau, and Alexandre Storelli. 2020. Enabling adaptive bitrate algorithms in hybrid CDN/P2P networks. In *11th ACM Multimedia Systems Conference (MMSys'20), June 8–11, 2020, Istanbul, Turkey.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3339825.3391859

1 INTRODUCTION

The ever increasing video traffic, according to Cisco studies, will account for 79% of global Internet traffic by 2020 [8], putting increasing pressure on video providers to provide good quality of experience (QoE) for their users who have different devices (smartphones, tablets, smart TVs, desktops...) and different bandwidth characteristics (WiFi, DSL, fiber...). Achieving a higher QoE for video streaming requires adapting the video playback to these variable viewing conditions; this has led to the emergence and the development of Adaptive Bitrate (ABR) streaming which has become the key technology for delivering video over the Internet. HTTP adaptive streaming relies on a web server to deliver media data. Media (audio, video) is encoded in multiple bitrates (qualities), and each quality is in turn subdivided into smaller parts, or segments, usually of constant duration but variable size. The choice of which segment to download is left to the client, which uses an online bitrate adaptation policy to maximize the QoE for the user. This is characterized by many contradictory decisions: downloading the highest possible bitrate, starting the video playback as fast as possible, keeping the switching rate between qualities reasonable while keeping the probability of re-buffering as low as possible.

Popular implementations of ABR streaming over HTTP are HTTP Live Streaming (HLS)[2] from Apple, Smooth Streaming (Smooth) [3] from Microsoft, Adaptive Streaming (HDS) [1] from Adobe and the MPEG-DASH [28] standard.

In addition, several ABR algorithms have been proposed. They differ in the required input information needed for the upcoming segment selection process. Based on this input information, ABR algorithms can be classified into three categories: (1) Buffer-based algorithms, like BBA [12] and BOLA [27], rely on the buffer characteristics, specifically buffer level, in their decision. (2) Throughput-based algorithms, such as Conventional, Panda [17], and Festive [14], observe the TCP throughput and adapt the bitrate accordingly. (3) Hybrid buffer-throughput algorithms, such as ABMA+ [4], use both throughput (segment download time) and buffer level in their decisions.

Once the ABR policy decision is made, the HTTP download request is typically sent to a Content Delivery Network (CDN).

Nowadays, thanks to their self-scaling properties, Peer-to-Peer (P2P) networks are becoming a popular alternative to CDN for delivering media content over the Internet. In P2P systems, each peer may ask for segments from either P2P network only or from both P2P and CDN (hybrid CDN/P2P) depending on the segment availability.

As a result of P2P and ABR improvements, there has been a lot of efforts to merge these two approaches and many questions have been raised. Since in ABR each client chooses each segment based on its current viewing conditions, different clients end up

MMSys'20, June 8–11, 2020, Istanbul, Turkey

ACM ISBN 978-1-4503-6845-2/20/06…$15.00
https://doi.org/10.1145/3339825.3391859

watching the video in different qualities, which makes the segments exchanging between users (peers) more difficult.

Interestingly, most of the existing ABR algorithms were designed to work in a server-based scenario, where the client requests segments directly from the CDN. However, when running these algorithms in a P2P network, all the estimated input information (throughput, segment download time, buffer value) will be dependent on the P2P connections of the client. Some related works implemented the ABR approach in P2P networks for both live [20] [24] and VoD [13] media streaming, however, no prior work focused on enabling the existing ABR algorithms in P2P systems.

In this paper we study the behavior of four state-of-the-art ABR algorithms in P2P pre-fetching environment, and comment on the main problems encountered. We also propose a new solution, called Response Delay, to make these algorithms compatible with the presence of P2P networks.

The rest of the paper is organized as follows. Section 2 provides an overview about existing work in this domain. Section 3 describes the P2P system model used. Section 4 presents the main challenges related to the current ABR algorithms with P2P, and Section 5, provides the proposed methodologies to enable the work of ABR algorithms in P2P networks. The methodologies are evaluated in Section 7, under simulation and realistic scenarios. Finally, in Section 8, the paper is concluded and future work is discussed.

2 RELATED WORK

2.1 HTTP adaptive streaming algorithms

In our study, we chose 4 common ABR algorithms that belong to two different classes (throughput-based and buffer-based). These algorithms have been studied and compared in CDN environments in the literature [15][29][27], but a study of these algorithms in the P2P-mesh environment is still missing.

2.1.1 Throughput-based Adaptation. Throughput-based adaptation algorithms, such as *Conventional* and *PANDA*[17], use only the TCP throughput measurements over enough probes as a mean to decide on the next segment bitrate. These algorithms differ mostly in the way they estimate and use the throughput. *Conventional*, *Panda*, use a four-step adaptation model: starting with estimating, then smoothing, and then quantizing the bandwidth and lastly scheduling the next segment. However, *Panda* uses a probing method similar to the TCP congestion control and has an additive-increase-multiplicative-decrease (AIMD), which makes PANDA more effective in terms of network utilization and fairness as users will compete less aggressively for network resources.

2.1.2 Buffer-based Adaptation. Buffer-based adaptation algorithms decide on the next segment bitrate based on the buffer characteristics mainly, as the bandwidth variations can be translated into changes in the buffer filling rate. Usually, these algorithms divide the buffer into different ranges and take different decisions for each buffer level range. *BBA* is one of the most well-known buffer-based algorithms. In [12] authors started the adaptation by mapping the buffer occupancy to a certain bitrate, then generalized the design to mapping the buffer occupancy to the segment size since the buffer dynamics are chunk size dependent. *BOLA* (Buffer Occupancy based Lyapunov Algorithm) [27] is an online control algorithm that uses Lyapunov optimization to achieve better QoE. It formulates the bitrate adaptation as a utility maximization problem; the utility increases by increasing the average bitrate, whereas rebuffering decreases it.

2.2 P2P System Architecture

P2P architectures can be classified into tree or multitree-based, mesh-based and hybrid tree-mesh based systems [5]. In tree-based architectures, such as ESM[10] and SplitStream[7], the peers are organized in tree-like layers, where each layer has one seeder and multiple leechers. The seeders flow the data to the leechers and organise the join/leave processes of their leechers. Although this architecture is easy to implement, it is vulnerable to peer churn.

To avoid this vulnerability, mesh-based architectures, such as CoolStreaming [32], and [19] were introduced. In mesh-based topology, even if there is a large size of peers, although each peer may have information about other peers in the system, yet it only connects to a small number of peers called peer pool. Each peer can exchange data with multiple peers inside this peer pool. Peers in mesh-based topology are self organised, and every peer can be seeder and leecher at the same time, which facilitates the peer-pool management and resiliency against random peers joining or leaving. However, the tradeoff is a higher network overhead due to exchanging more control messages within the peer pool.

The hybrid tree-mesh based schemes, like MultiPeerCast [18] and LayeredCast [21], inherits the benefits of both tree and mesh architectures. In this scheme, some children can get data from different parents (mesh structure), whereas other children can get the data from only one parent (tree structure). However, most of the hybrid schemes still face the complexity of the tradeoff between stability and scalability [11].

In this work, we will use a mesh-based P2P protocol for both simulation and real measurements.

2.3 Hybrid CDN/P2P video streaming

Due to the complementary advantages of CDN being reliable, and P2P being cheap and scalable, a system which combines both technologies can be highly beneficial [31] [30]. Such a system includes three main components: (1) the actual media server who distributes the video content to the clients. (2) A set of clients who are watching the same video content, and (3) a tracker who finds the best peers by matching the clients who are watching the same video content, on the same quality level, if possible, and in adjacent time windows. Authors in [9] propose such a hybrid system for live video streaming over the Web, to reduce the CDN usage as much as possible. In short, the video segments are portioned further into small chunks of roughly equal size. Besides the video player module, which consecutively requests video segments to fill the playout buffer, clients also have an additional P2P module which pre-fetches the video chunks ahead of time. However, the pre-fetching process of one segment may be incomplete due to the peers' heterogeneity, and thus, a cost-effective solution is to download the missing data from CDN only. A similar architecture will be used in this work, as provided later in Section 3.

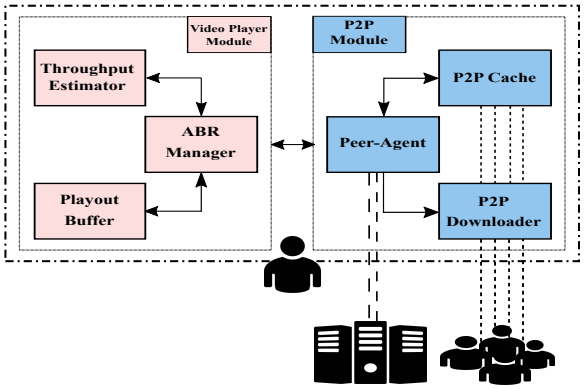


Figure 1: Hybrid CDN/P2P HTTP adaptive streaming

2.4 QoE and P2P efficiency evaluation

The quality of experience (QoE) is reported in [22] and [26] to be high influenced by the perceived video quality, the frequency of quality switches and the video playback interruptions. To meet the QoE specifications, many metrics were proposed in different related works, to assess the performance of the ABR algorithms, not only in CDN networks [17][4][16], but also in P2P [25] networks.

Furthermore, it should be noted that ABR streaming in hybrid CDN/P2P may induce an inefficient usage of the resources and, as a result, data overhead. Some of this overhead comes from the ABR policy switching the quality of the next segments, while having these segments already pre-fetched but in different quality, hence not used by the player. Another form of the overhead happens when a partial P2P segment is completed by a CDN request but P2P chunks for this segment are still received (no or late canceling of scheduled chunks). Finally, a third form of overhead may also happen in partial P2P segment case if the CDN replies to the completion byte-range request with a larger byte-range, as can be seen in the real world; we, however, ignore this last form of overhead in our simulations.

The overhead is evaluated as an important metric in different prior works. Authors in [6] address the overhead resulting from the multiple-source adaptive streaming. However, their main concern is the overhead coming from multiple description coding (MDC) compression scheme, which we do not cover in this work. In [24] authors use cache *hit* and cache *miss* ratios to measure the overhead. Their main metric, *traffic savings*, reports the amount of P2P data from the total data consumed by the peers. However, they do not measure the overhead per peer, which is the amount of P2P data not used by the peer itself, or not used by other peers in the peer-pool. Our work focuses on these measures of ABR-related overhead, through two proposed metrics in Section 6.2.

3 SYSTEM MODEL

In this section, we describe the hybrid CDN/P2P model. Table 1 summarizes the notations used in this paper.

In this system, as shown in Figure 1, peers have two main modules: the video player and the P2P modules. This work deliberately isolates both modules, to be able to test any player logic over P2P network, but without modifying the ABR or player logic.

Table 1: Notation

Notation	Explanation
N	Total number of the player requested segments
K	Total number of the P2P pre-fetched segments
Q_n	Buffer level after adding segment n (s)
$t_{dw}[n]$	Video segment download time
$bw[n]$	the player measured bandwidth of segment n
bw_{cdn}	CDN measured bandwidth
bw_{p2p}	P2P measured bandwidth
$r[n]$	Bitrate of segment n (Mbps)
τ	Video segment duration (s)
t_{rqst}	ABR scheduling segment time
δ	Segment fetching time from P2P cache
$S[n]$	Size of segment = $s[n]_{p2p} + s[n]_{cdn}$
$s[n]_{cdn}$	Downloaded data from CDN
$s[n]_{p2p}$	Downloaded data from P2P
t_{cdn}	Download time from CDN
t_{p2p}	Download time from P2P
d_n	Segment Response delay
$st[n]$	Cached segment state

Algorithm 1 video player logic

1: $initialize(Q_n, t_{dw}[n]) \leftarrow 0$ for n =1
2: **for** n in $[1, N]$ **do**
3: $(r[n],\ t_{rqst}) = ABR(Q_{n-1},\ t_{dw}[n-1],\ bw[n-1])$
4: $t = t + t_{rqst}$
5: $Q_n = Q_{n-1} - t_{rqst}$
6: $S[n] = PeerAgent(n, r[n])$
7: $bw[n] = S[n]/t_{dw}[n]$
8: $t = t + t_{dw}[n]$
9: $Q_n = Q_n + \tau - t_{dw}[n]$
10: **end for**

The video player requests the video segments in their playback order and fills the playout buffer. A simple pseudo-code of the used player logic is shown in Algorithm 1. It runs an ABR algorithm to decide on the bitrate $r[n]$ and the request scheduling time t_{rqst} of the next segment to download (line 3 of Algorithm 1). The request is then forwarded to the peer-agent which delivers back the segment $S[n]$ after the download time $t_{dw}[n]$ (line 6 of Algorithm 1). Simply, the bandwidth is measured as the downloaded data over its download time and the simulation time t increases by this download time (lines 7 and 8 of Algorithm 1). The playout buffer Q_n consumes some data while downloading the segment, then grows by one segment duration once the segment is buffered (line 9 of Algorithm 1).

The segment requests are forwarded to the P2P module, which is composed of three main sub-modules: peer-agent, P2P downloader and P2P cache. ***The peer-agent***, as indicated in Algorithm 2, receives the requests for the video segments at specific qualities.

Algorithm 2 peer-agent logic

1: $n \leftarrow$ the player requested segment
2: $st[n] \in \mathbb{S} : \mathbb{S} = \{AC, A\overline{C}, \overline{A}\} \leftarrow$ cached segment state
3: **if** $st[n] = AC$ **then**
4: **if** time needed before delivering the segment **then**
5: $t_{dw} = d_n$ ▷ Section 5: Response-Delay
6: **else**
7: $t_{dw} = \delta$
8: **end if**
9: **else if** $st[n] = A\overline{C}$ **then**
10: $dataRange = S[n] - downloadedData[n]$
11: $sendCdnRequest(dataRange)$
12: **if** time needed before delivering the segment **then**
13: $t_{dw} = t_{cdn} + d_n$ ▷ Section 5: Response-Delay
14: **else**
15: $t_{dw} = \delta + t_{cdn}$
16: **end if**
17: **else if** $st[n] = \overline{A}$ **then**
18: $sendCdnRequest(S[n])$
19: $t_{dw} = t_{cdn}$
20: **end if**
21: wait for t_{dw} to get and deliver the segment
22: **return** $S[n]$

It then checks the segment availability in the P2P cache. These segments have three possible states: available and completed (AC), available but not completed ($A\overline{C}$) or not available ($\overline{A}$). If the requested segment is available and completed, the peer-agent delivers it back in a very short time δ (line 7 of Algorithm 2), which is the time needed to fetch the segment from the P2P cache. Otherwise, peer agent sends CDN requests either to download only the missing range of data if the segment is not completed yet (lines 10 and 11 of Algorithm 2), or to download the whole segment if it is not available in the P2P cache (line 18 of Algorithm 2). It then delivers the segment to the video player right after time t_{cdn} which is the time it took to download the segment (or parts of) from CDN (lines 15 and 19 of Algorithm 2).

The P2P downloader keeps downloading data from peers and filling the P2P cache. This work is divided into two processes: scheduling and fetching. In the scheduling process, for every time window w, the P2P downloader handles three main tasks: choosing segments to schedule first, selecting seeders for these segments, and assigning chunks of data to each seeder.
To handle the first task, line 1 of algorithm 3, the P2P downloader schedules a list of ($A\overline{C}$) or ($\overline{A}$) consecutive segments of the same quality level as the last requested segment by the video player. The second task, which is peers selection (line 2 of algorithm 3), is handled first by checking the peers who have already downloaded the required segments and whose estimated upload capacity is high enough to download the segments. The last step in the scheduling phase, line 3 of algorithm 3, is assigning the chunks of the segments to be played first to the peers who are selected as best seeders. The received chunks are stored in the ***P2P cache*** which can store up to 200 MB of data in our model. The maximum cache size is chosen to be similar to web browsers maximum cache sizes, and the cached segments are organised and manipulated using a FIFO (first in first out) list.

Algorithm 3 P2P downloader logic

1: segmentsToFetch = updateSegmentsFetchingList(n,r[n]).
2: seeders = updateSeeders(segmentsToFetch).
3: chunks = assignDataToSeeders(seeders, segmentsToFetch)
4: sendP2PChunksRequests(chunks, seeders)
5: saveFetchedDataInP2PCache(FetchedData)

4 CHALLENGES OF ABR ALGORITHMS IN P2P NETWORKS

When deployed over a hybrid CDN/P2P network, classic ABR algorithms face the following problems:

- In throughput-based ABR algorithms, the next download decisions are taken based on the most recent bandwidth estimation. However, the P2P network conditions vary a lot during the streaming session due to the high dynamics and heterogeneity of the peers. Also, cached (pre-fetched) P2P segments are delivered almost instantaneously, resulting in a very high bandwidth estimation by the ABR algorithm. As a result, such algorithms will end up selecting the highest bitrate for the next segments, which in turn makes the ABR more fragile to any upcoming bandwidth fluctuation, resulting in well-known quality oscillations.
- The buffer-based ABR algorithms, as mentioned, rely on the buffer occupancy to decide on the next bitrate to download. In P2P, the buffer fill rate will vary a lot, depending on the segment source (pre-fetched, from peers only, from CDN only or from peers and CDN); this induces more frequent changes in buffer level estimation, therefore leading to more quality switching.
- Finally, the pre-fetching model downloads the next segments on the same quality as the last requested one. With an ABR algorithm changing the current quality too often, the pre-fetched segments are more likely to be unused because not in the desired quality, thereby diminishing the P2P efficiency.

5 PROPOSED SOLUTION: RESPONSE-DELAY

The main issue of the various ABR algorithms being confused by the presence of P2P cached segments can be addressed by making the ABR believe that such segments were downloaded from CDN. Unfortunately, tracking the CDN bandwidth variation can only be done when the segments are requested and downloaded from CDN, which is not the case when the P2P network is active. In this paper, we show that this issue, regardless of the type of ABR algorithm, can be solved by adding the right delay to the responses.

5.1 Principle

Since our design goal is to keep the ABR and video player logic unmodified and agnostic of the P2P network, it is not possible to directly control the ABR algorithm; we therefore only influence the algorithm by adding a delay before returning the requested segment. This additional delay will be interpreted as a longer download time

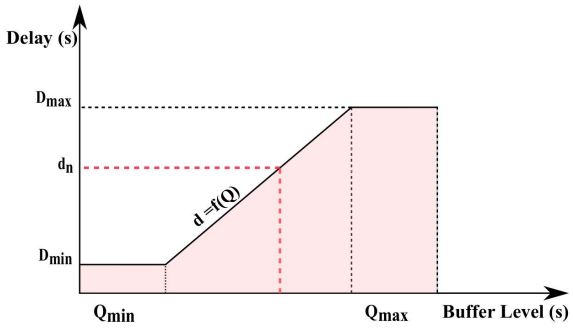


Figure 2: The buffer-delay map used for response delay

for the segment and by modulating it, we change the effective download time as seen by the ABR logic. Acting as a replacement for the HTTP stack of the player, Response-Delay has the advantage of being generic and works with any video player and most ABR algorithms.

Response-Delay is designed to prevent the undesired network variation feedback, which results in uncontrolled ABR decisions. Our goal then is to add a good delay that should: (1) Eliminate the re-buffering events that occur due to the wrong P2P speed estimation. (2) Reduce the number and the amplitude of the quality switches when P2P is applied. (3) Keep a reasonably high average quality. (4) Increase the P2P percentage. (5) Reduce the number of unused P2P downloads due to pre-fetching.

However, choosing the appropriate response delay is a question of compromise. Too long, on one hand, can starve the player buffer and lead to undesirable rebuffering events. It also makes the ABR falsely believe the available bandwidth is low and switch to lower content qualities. Too short, on the other hand, leads the ABR switch to excessive qualities that the available bandwidth cannot support, leading to further rebuffering events.

5.2 Response-Delay proposals

Inspired from the two main classes of ABR algorithms, we propose a buffer-based and a network-based approach.

5.2.1 Buffer-delay map (BufDel). Inspired from the buffer-based ABR, where the bitrate is selected from lowest to highest as the buffer increases from low level to the maximum level, we propose to use a continuous function that increases the delay as a function of the buffer occupancy. We introduce BufDel, a buffer-based response delay, that computes a delay $d = f(Q)$ as shown in Figure 2. This delay speeds up the filling of the buffer with segments when the buffer level is low, whereas it slows it down when the buffer level is evolving to reach the maximum target. As discussed, the delay should be bounded, therefore $d_n = f(Q_n) \ni Dmin < d_n < Dmax$ and $Qmin < Q_n < Qmax$. The boundaries are chosen to be $Dmin > 0$ and $Dmax = \frac{s[n]_{p2p}}{r[n]} \leq \tau$. BufDel requires monitoring of the buffer level of the player, which in Web environment is done by monitoring the <video> element without modifying the video player.

5.2.2 Network delay (NetDel). In this approach, the delay is used to modify the current segment download as seen by the ABR and video player logic, and the response delay is computed based on the available bandwidth measurements. However, these measurements mix the contributions of CDN and P2P links; summing them is not appropriated as CDN and P2P traffic are generally not simultaneous. In our model, CDN is only used when P2P cannot support the bitrate. The most appropriate way to determine the available bandwidth in this context is therefore to take the highest bandwidth between CDN and P2P. This approach has the advantage to be more resilient to obsolete measurements of CDN bandwidth. Note that measurements of CDN bandwidth are not taken into account in cases of small byte-range requests, as they are usually unstable and inaccurate in that situation. This approach works as a CDN/P2P link switcher targeting the highest measured bandwidth (1), and thus driving the ABR to pick a higher quality that can be sustained via the highest bandwidth.

$$targetBw = \max(bw_{cdn}, bw_{p2p}). \quad (1)$$

The delay is calculated as shown in (2), such that the segments with different sizes will be delivered with different delays; this ensures that the ABR detects that bigger segments need more time to be downloaded than shorter ones for the same quality. Additionally, the segment delay is upper bounded to the segment duration, otherwise, undesired playback pausing may occur while waiting for the segment to be delivered.

$$d_n = \frac{s[n]_{p2p}}{targetBw} \leq \tau \quad (2)$$

Moreover, in a scenario where the last P2P and CDN throughput measurements are lower than the bitrate of the next pre-fetched segment, the ABR may estimate a lower bandwidth according to (1), and may down switch the current quality. If the next segment is pre-fetched in higher quality, it is preferable to prevent the quality down switch and try to stay at the same quality by controlling the response delay to (3).

$$d_n = \frac{s[n]_{p2p}}{\max(targetBw, r[n])} \quad (3)$$

5.3 Applying Response-Delay

In this section, we discuss when to use the Response-Delay, or more precisely which segments should be delayed. Obviously, all the *AC* P2P segments (completed in P2P cache at the time of the request by the ABR) should be delayed before being sent to the player (line 5 in Algorithm 2). Similarly, CDN segments (nothing pre-fetched) are already delayed by the speed of CDN connection and will be delivered immediately once they are downloaded (line 19 in Algorithm 2). But hybrid segments, which are downloaded from both CDN and P2P, are handled differently (line 13 in Algorithm 2). Peer-Agent will wait for a time t_{cdn} to have these segments completed before delaying the P2P part with the time d_n.

6 EXPERIMENTAL EVALUATION

6.1 Experimental Setup

In this work, we used MATLAB for building the model described in Section 3. We simulate one peer pool of 10 peers, which is a good representative of the actual peer pool size for mesh-based systems.

Table 2: Parameters used for the ABR algorithms

Algorithm	Parameter	Value
BBA	r	11.25
	cu	15.75
BOLA	γp	5
	V	2.012
PANDA	α	0.2
	ϵ	0.15
	κ	0.14
	ω	0.3
	β	0.2
	B_{min}	26
CONV	α	0.2
	ϵ	0.15

They share the same video content of 10-min long, segmented in 299 segments of 2 seconds length, and encoded in 6 different bitrates: 0.59, 1.032, 1.54, 2.13, 3.078, and 4.219 (Mbps). For the video player parameters, we set the maximum buffer size of 30 seconds, and the playback rate is set to one (nominal playback speed). We set both of the play-back startup and the re-buffering thresholds to one video segment. Every 15 seconds, a new peer joins the session. The first peer starts the session by connecting to the CDN, while other participating peers are connected on both CDN and P2P links. For a fair comparison, all peers experience the same upload and download bandwidth conditions, and they all initiate the session from the same starting point (identical network throughput at first segment request). To evaluate our proposal, we use in Section 7.5 some publicly-available 3G sets of real bandwidth traces [23]. We used these traces since they have been used a lot in the literature to study and compare the used ABR algorithms. As shown in Figure 3, we chose the traces that show the normal bandwidth variations and fewer outages duration, corresponding to direct throughput measurements from a bus, a train, and a car. Also, to shed the light on some different ABR challenges, we use some controlled bandwidth traces, as later shown in sections 7.1 to 7.4.

6.2 Evaluation Metrics

We first investigate the impact of our methodology on QoE using the suggested QoE metrics in [16]. The *average quality* is the average selected bitrate of the video segments. The *stability* and the *smoothness* metrics show how often the ABR switches between different qualities and the amplitude of these switches. The streaming *continuity* and *consistency* metrics report the number and the duration of the video playback interruptions.

Beside QoE metrics, we evaluated the overall P2P system performance in terms of the efficient usage of the P2P and CDN resources. A good P2P system would be able to reduce the CDN requests and the P2P overhead as much as possible. We first introduce the metric *P2P Offload*, as calculated in (4), which indicates the cost-effectiveness in terms of the CDN usage; i.e. the less data

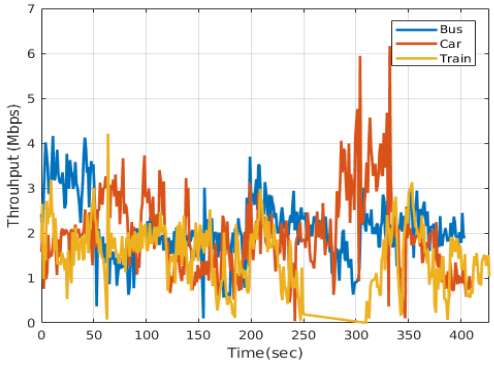


Figure 3: Example of the selected bandwidth profiles

downloaded from CDN, the better P2P offload.

$$P2POffload = 1 - \frac{1}{N}\sum_{n=1}^{N}\frac{s[n]_{cdn}}{S[n]} \tag{4}$$

As previously mentioned, the inaccurate pre-fetching process results in inefficient usage of P2P resources. P2P segments, denoted as K, may be pre-fetched in m different qualities. However, only one quality will be useful and will be requested by the player, while the other $m-1$ qualities are overhead and useless for the peer. Nevertheless, some of these unused segments might be requested by other peers, and the rest is not used by any of the peers. We note $s[k]_{p2p}$ the P2P data of the useful quality, $s[k]_{\overline{p2p}}$ the P2P data of all the useless qualities for one peer that are used by peers and $s[k]_{\overline{\overline{p2p}}}$ the P2P data of all the useless qualities of one peer that are unused by any of the peers. Therefore, the total P2P data per segment k is measured as it is shown in (5).

$$P2P[k] = s[k]_{p2p} + s[k]_{\overline{p2p}} + s[k]_{\overline{\overline{p2p}}} \tag{5}$$

The Peer Efficiency, as calculated in (6), reports the average useful P2P data over the total P2P data of K segments. And the last P2P metric, Peer-Pool-Efficiency, is the average of the reused P2P data over the total P2P data, for K P2P segments as shown in (7).

$$PeerEfficiency = \frac{1}{K}\sum_{k=1}^{K}\frac{s[k]_{p2p}}{P2P[k]} \tag{6}$$

$$PeerPoolEfficiency = \frac{1}{K}\sum_{k=1}^{K}\frac{s[k]_{\overline{p2p}}}{P2P[k]} \tag{7}$$

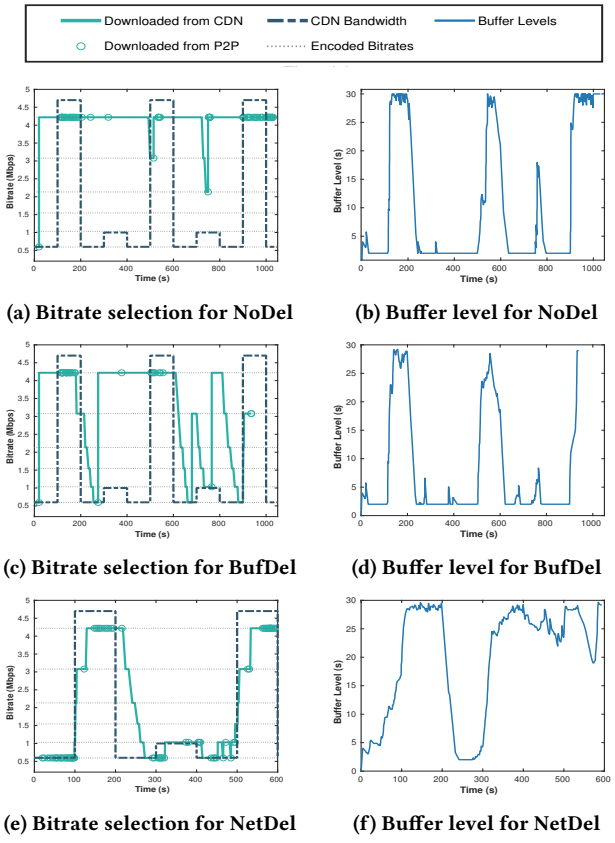
7 RESULTS AND DISCUSSIONS

In this section, we evaluate the performance of four state of the art ABR algorithms: *BBA*, *BOLA*, *PANDA* and *CONVENTIONAL* (denoted as *CONV*). For each algorithm we used the default suggested parameters as shown in Table 2. The used scenarios are: *CDN-only* is the normal CDN-based streaming with no P2P streaming, *NoDel* is a normal hybrid CDN/P2P streaming without applying any of the Response-Delay methods. *BufDel* is a hybrid CDN/P2P scenario using the BufDel approach described in part 5.2.1 and the

last scenario *NetDel* is a hybrid CDN/P2P scenario using the NetDel method described in part 5.2.2.

7.1 Non-conservative throughput-based algorithms

The non-conservative throughput-based ABR algorithms, e.g. CONVENTIONAL, choose the bitrates aggressively by following the measured bandwidth closely.



(a) Bitrate selection for NoDel (b) Buffer level for NoDel

(c) Bitrate selection for BufDel (d) Buffer level for BufDel

(e) Bitrate selection for NetDel (f) Buffer level for NetDel

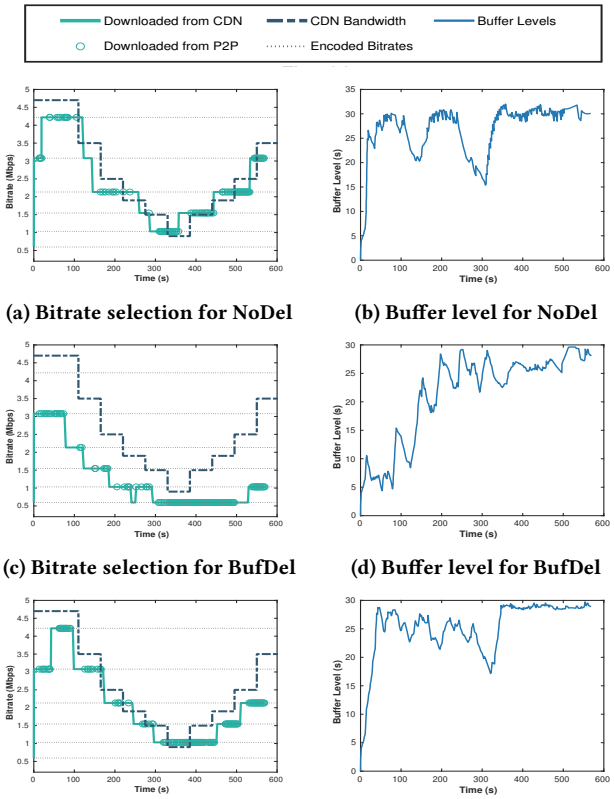
Figure 4: Conventional's bitrate selection and buffer level for NoDel, BufDel, NetDel scenarios

Looking at Figure 4 which compares the bitrate selection and the buffer levels of CONVENTIONAL for the *NoDel*, *BufDel* and *NetDel* scenarios, we see that for NoDel scenario, whenever a P2P segment is fetched from P2P cache, the ABR overestimates the bandwidth. Therefore, it stays on the highest quality for some time before re-adapting again. Meanwhile, if the current bandwidth can not sustain this quality, as shown in Figure 4a, the playout buffer depletes to a low level causing re-buffering, as clearly shown in Figure 4b. Unfortunately, for BufDel, the same problem repeats, but only when the buffer level is low, and some P2P segments are fetched from the P2P cache (looking at 10s, 270s in Figure 4c and Figure 4d): these segments are still delivered fast, causing the same problem as previously discussed with NoDel. Interestingly, NetDel avoids this issue by adapting to the actual bandwidth, as shown in Figure 4e. NetDel is also more efficient in terms of P2P usage by giving the system more time to pre-fetch data and complete segments from P2P, which is not the case for BufDel when the buffer level is low (fast response time hence less time to complete pre-fetch of next segment).

Also, it should be noticed that when the rebuffering occurs more often, it causes longer playout time, as it is seen when looking at the x-axis of figures 4a, 4b, 4c and 4d (1000s nearly) compared to the one from figures 4e and 4f (600s).

7.2 Conservative throughput-based algorithms

The conservative throughput-based algorithms, such as PANDA, adapt to the estimated bandwidth gradually, i.e. wait for some time before switching the video quality.



(a) Bitrate selection for NoDel (b) Buffer level for NoDel

(c) Bitrate selection for BufDel (d) Buffer level for BufDel

(e) Bitrate selection for NetDel (f) Buffer level for NetDel

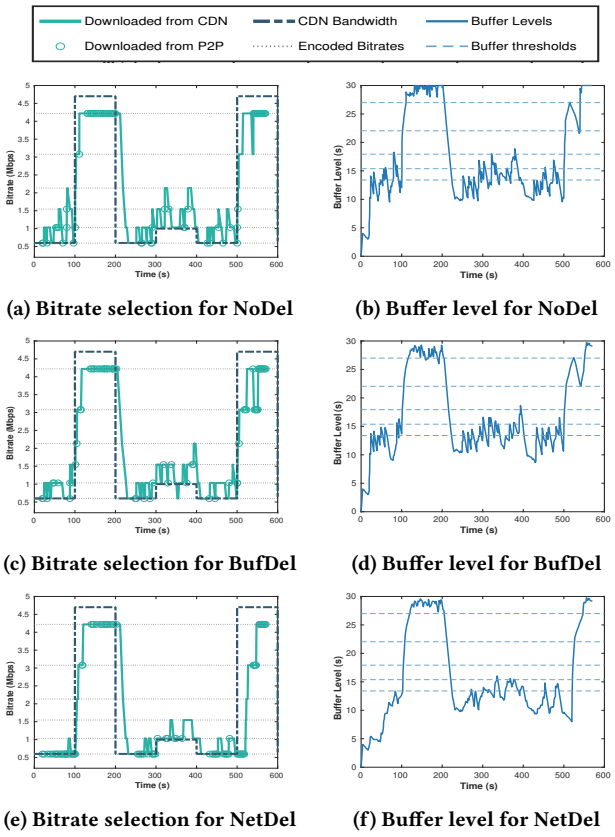
Figure 5: PANDA's bitrate selection and buffer level for NoDel, BufDel, NetDel scenarios

When it comes to NoDel, PANDA shows to be resilient even when P2P segments are delivered fast. It gradually smooths the bandwidth estimation and adapts accordingly (see Figure 5a, 5b). However, BufDel brings a different problem in terms of bandwidth underestimation. The quality is clearly lower when looking at Figure 5c, in particular when many consecutive P2P segments are delivered nearly at the same average bitrate: PANDA loses tracking of the actual bandwidth and requests the same quality for a long time, until receiving some CDN segments. This problem may also happen with the non-conservative throughput algorithms, however,

they recover faster whenever a new CDN segment is downloaded. This problem has less influence when applying NetDel. The only drawback of this technique is its relying on the last CDN measurements using (1). In a scenario (as the one shown in Figure 5e) where the measured P2P bandwidth and the last measured CDN bandwidth are both low, NetDel will target a low bandwidth, making the ABR lowers the quality as well. This behavior persists as long as the current P2P bandwidth is low, and the requested segments are delivered from P2P cache.

7.3 Buffer-based algorithms

We extend our evaluation to buffer based algorithms, illustrated with BBA in this section.



(a) Bitrate selection for NoDel (b) Buffer level for NoDel

(c) Bitrate selection for BufDel (d) Buffer level for BufDel

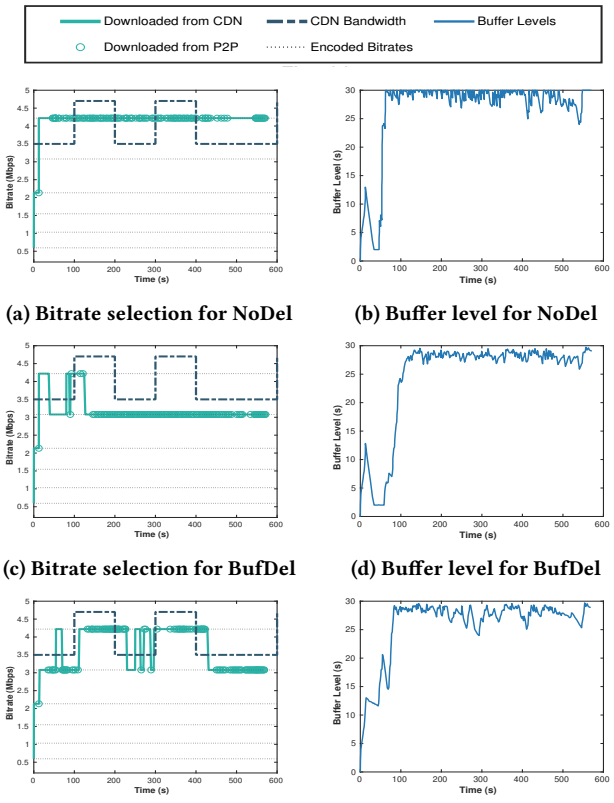
(e) Bitrate selection for NetDel (f) Buffer level for NetDel

Figure 6: BBA's bitrate selection and buffer level for NoDel, BufDel, NetDel scenarios

With NoDel, BBA is shown to be less influenced by the rebuffering (see Figure 6b) since it adapts to the buffer occupancy directly. But when looking at Figure 6a, another interesting issue of frequent quality switches appears. In that scenario, the BBA buffer thresholds are 13.4, 15.38, 17.93, 22.04 and 26.99 seconds. At time 23.1s the buffer level is 10.8s, the player receives 4 seconds of two consecutive P2P segments of 0.59Mbps. The buffer then grows to 14.8s, crossing the threshold at which the player switches to 1.032 Mbps. The next segment of 1.032 Mbps is not available in P2P cache, so it is requested from CDN, and it takes almost 3.9s to be downloaded. Meanwhile, the buffer depletes to 10.9s, crossing the threshold and switching back to 0.59Mbps. Then at 27s, it again receives another two P2P segments and switches up to 1.032 Mbps. Thus, the cycle repeats whenever P2P segments are received fast while the actual bandwidth can not sustain the up switches. Unfortunately, BufDel does not help in avoiding these oscillations, as shown in Figure 6c. Indeed, when BufDel detects a low buffer level, it accelerates the delivery of P2P segments, making the buffer level cross the switching thresholds back and forth again. Contrary to NoDel and BufDel, NetDel copes with this problem by adjusting the delivery speed of P2P segments to the current bandwidth measurements, but at the cost of a lower bitrate (see Figure 6e and 6f).

7.4 Results with normal high network profiles

To gain more insightful results, we also evaluate the behavior of our proposals with a normal high network profile where the bandwidth conditions are not so extreme and high enough to sustain at least the highest two qualities.



(a) Bitrate selection for NoDel (b) Buffer level for NoDel

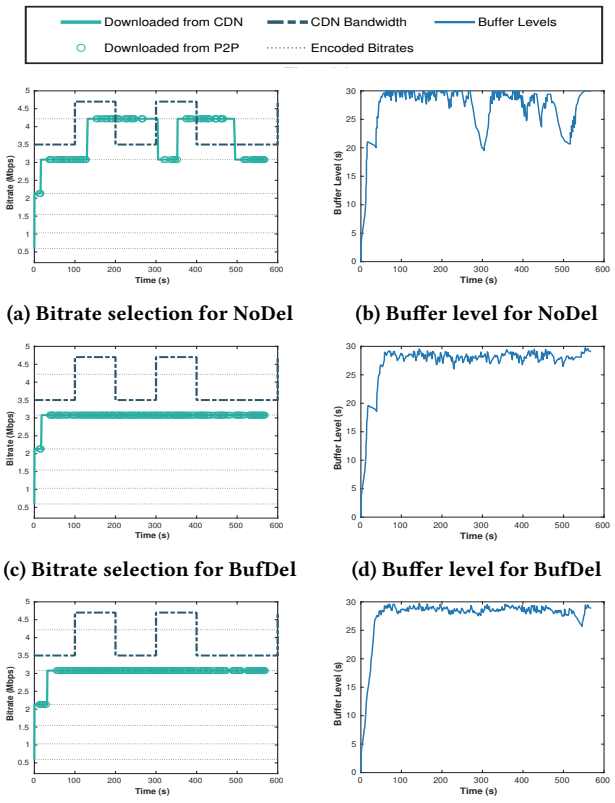
(c) Bitrate selection for BufDel (d) Buffer level for BufDel

(e) Bitrate selection for NetDel (f) Buffer level for NetDel

Figure 7: CONV's bitrate selection and buffer level for NoDel, BufDel, NetDel scenarios with a normal high trace

Starting with CONV algorithm, the fast delivery of P2P segments does not seem to be a problem with the high bandwidth, see Figure 7a and Figure 7b, it rather improves both of the QoE and the P2P efficient usage. The problem of BufDel leading the ABR to select the same quality for long time, reappears obviously in Figure 7c

again as long as the P2P segments are pre-fetched and delivered at nearly the same average bitrate. With NetDel, looking at Figure 7e, we can see that the bitrate selection follows the bandwidth variations closely, sacrificing the QoE in terms of less average rate and a higher track switches comparing to NoDel scenario.



(a) Bitrate selection for NoDel (b) Buffer level for NoDel

(c) Bitrate selection for BufDel (d) Buffer level for BufDel

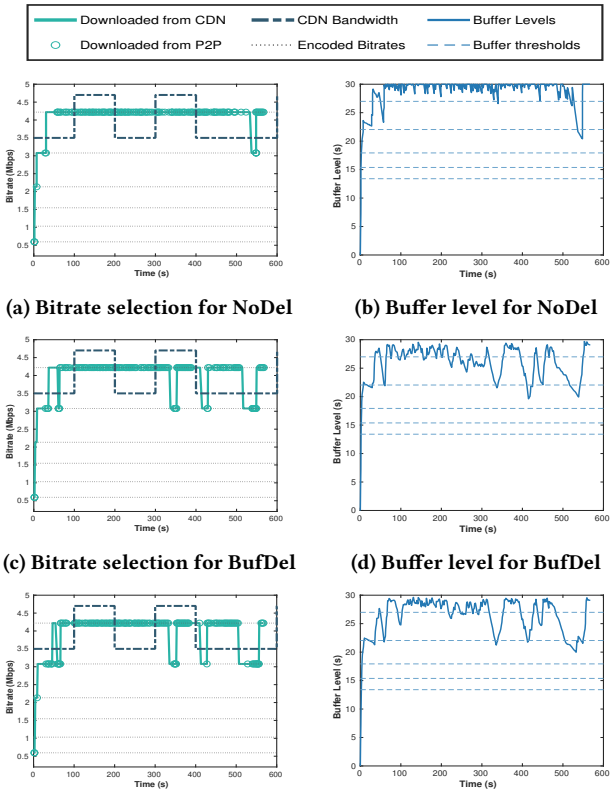
(e) Bitrate selection for NetDel (f) Buffer level for NetDel

Figure 8: PANDA's bitrate selection and buffer level for NoDel, BufDel, NetDel scenarios with a normal high trace

For PANDA, the bitrate selection (Figure 8c and 8e) shows that the response delay does not affect the conservative adaptation of PANDA, whereas with NoDel, the bandwidth overestimation breaks this conservative approach. Finally, we can see that BBA with NoDel manages to keep the maximum buffer level by receiving the P2P segments fast, and therefore stays at the highest quality for a longer duration compared to BufDel and NetDel, for which the delayed delivery of P2P segments results in consuming more buffer and therefore switching to lower qualities.

7.5 All metrics evaluation

We further evaluate the performance of the two proposed approaches over the combination of QoE and P2P metrics introduced in Section 6.2. All the results are averaged over all peers and all different 3G traces except for P2P metrics, for which the first peer, which is connected to CDN link only, was excluded.



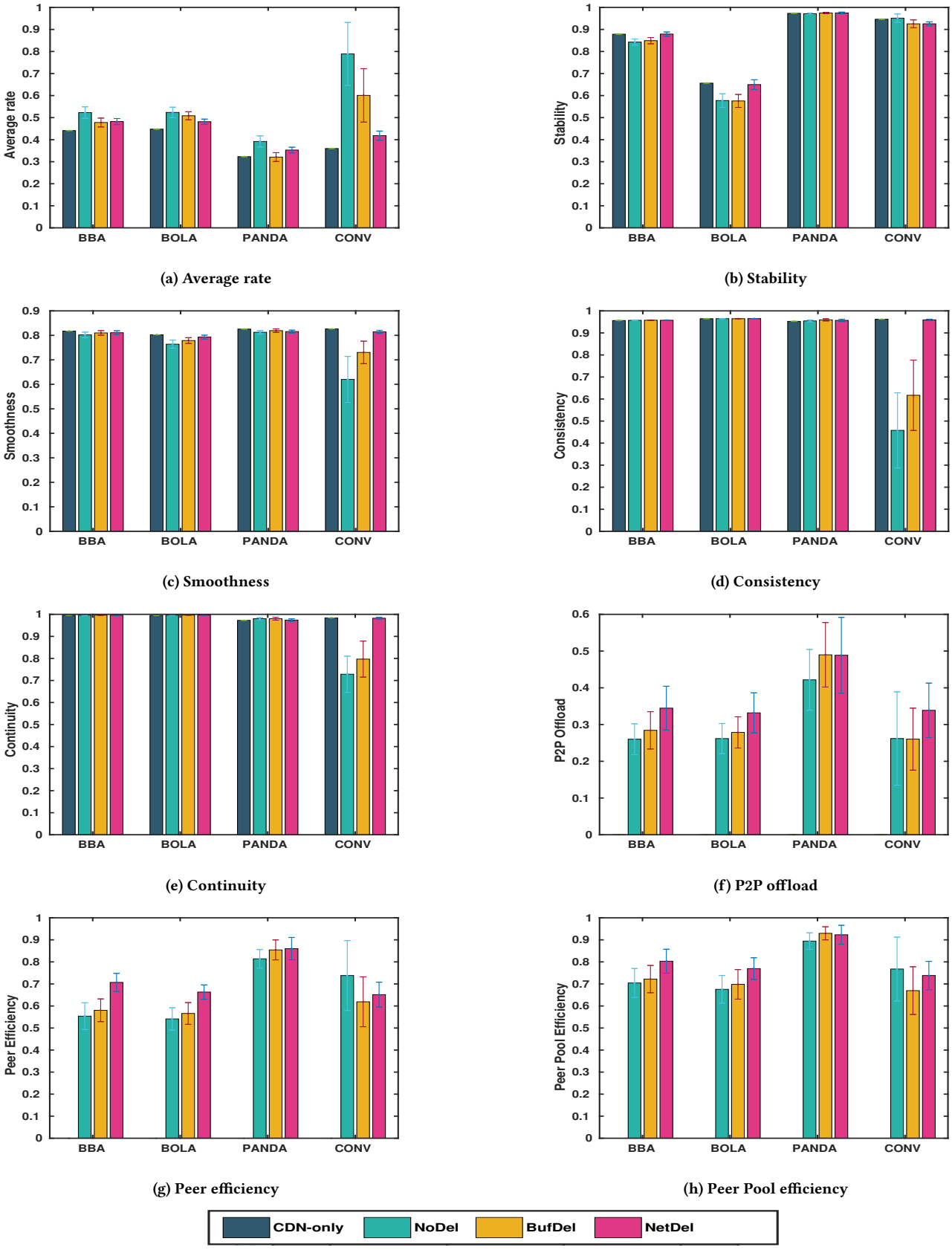
(a) Bitrate selection for NoDel (b) Buffer level for NoDel

(c) Bitrate selection for BufDel (d) Buffer level for BufDel

(e) Bitrate selection for NetDel (f) Buffer level for NetDel

Figure 9: BBA's bitrate selection and buffer level for NoDel, BufDel, NetDel scenarios with a normal high trace

7.5.1 QoE metrics. Starting with the average rate metric, shown in Figure 10a, both approaches improve the average quality compared to the CDN-only scenario. As expected, BufDel gains more average quality compared to NetDel for buffer-based algorithms (BBA and BOLA) and non-conservative throughput algorithms (CONVENTIONAL) because of its effects on the responsiveness of these algorithms (as documented in sections 7.1 and 7.3). Regarding the stability (the number of quality switches), Figure 10b, NetDel achieves the same stability as the one achieved by CDN-only for all algorithms but CONVENTIONAL, which is more sensitive to the bandwidth changes. The same result is observed for smoothness, as shown in Figure 10c: the transitions between the video qualities are almost as smooth as those using CDN-only, whereas NetDel is slightly better than BufDel since the latter still faces the fast delivery and the bandwidth over-estimation issue, in particular when the buffer level is low. For consistency, Figure 10d, all algorithms for all scenarios gain the same score except for CONVENTIONAL, where NetDel registers a significant improvement (up to 55%) to the normal P2P scenario; this is mostly gained from resolving the bandwidth overestimation issue. The same results are observed for continuity (number of the re-buffering events), Figure 10e, with up to 30% improvement for CONVENTIONAL when applying NetDel compared to NoDel.



(a) Average rate

(b) Stability

(c) Smoothness

(d) Consistency

(e) Continuity

(f) P2P offload

(g) Peer efficiency

(h) Peer Pool efficiency

Figure 10: Results for QoE and P2P metrics with 95% confidence intervals using the 3G network profiles

7.5.2 P2P metrics. Regarding the P2P metrics, NetDel shows higher improvement on the P2P Offload, as shown in Figure 10f, compared to NoDel and BufDel. Looking at Figure 10g, we can see that both BufDel and NetDel have better results on the peer efficiency compared to NoDel for BBA, BOLA and PANDA, with NetDel being the best. However, for CONVENTIONAL this metric is lower compared to NoDel; this is expected for NoDel since CONVENTIONAL does not switch often between qualities but it rather stays longer at the highest quality, leading to less quality switches-related overhead, which in turn means a higher peer efficiency. The same result is observed for peer pool efficiency, Figure 10h, where NetDel shows a better sharing of the overhead segments to other peers for all the ABRs except CONVENTIONAL (again, for the same reason of less overhead due to quality switches).

7.6 Commercial Service Trials

To verify our work with unknown ABR algorithms, we tested the two proposed methodologies in existing commercial streaming services, thanks to STREAMROOT technology providing the P2P backend. The trials were conducted with various HTML5 video players implementing their own, different ABR algorithms. The scenarios CDN-only and NoDel were omitted in these trials because of their bad P2P efficiency and QoE drawbacks on the users and the customers, they are not suited for commercial services. We therefore only compare the BufDel and NetDel with what we observed in our simulations. We launched the test for one day, with an overall range of 5k to 15k concurrent peers participating. The service provider used 3 different live streams, segmented into 10-second segments, and encoded in a different set of bitrates: [2.2, 1.2, 0.94, 0.446], [1, 0.796, 0.446] and [1.248, 0.698, 0.348] (Mbps). In this comparison, and to ease the data collection, we used some common production metrics, collected per 2 minutes intervals, and averaged over the whole session. These metrics are: the average rate in Mbps, the average number of quality switches per minute (CPM), the average number of experienced re-buffering events per minute (CPM), the average re-buffering duration, the ratio of the time spent on the maximum quality over the whole session (TRmax) and the ratio of the useful P2P data over the total downloaded data. Table 3 shows the comparison of BufDel and NetDel regarding these metrics. The overall results show that NetDel is slightly better in terms of quality switches and P2P efficiency. It also has a lower re-buffering duration but with a slightly higher number of re-buffering events. On the other hand, BufDel increases the average bitrate (per minute), and it stays longer on the highest quality. This is consistent with our simulation results.

Table 3: Commercial Service Trials

Metric	Method	
	BufDel	NetDel
Avg rate (Mbps)	1.183	1.173
Avg track switches (CPM)	0.117	0.115
Avg rebuffering events (CPM)	0.088	0.094
Rebuffering Duration (s)	6.75	6.43
TRmax %	88.90	87.29
P2P offload %	46.50	46.55

8 CONCLUSION

In this work, we discussed the main problems related to the usage of existing ABR algorithms in hybrid CDN/P2P networks. We also proposed Response-Delay, a novel algorithm ensuring the compatibility of existing ABR algorithms with P2P networks. We evaluated the performance of two methods, BufDel and NetDel, using four state-of-the-art ABR algorithms, over a set of network profiles. We also introduced two new metrics to quantify the P2P efficiency of our proposal. Finally, we tested our proposal on unknown ABR algorithms in a realistic scenario. Our results show that our proposal enables the work of these different ABR algorithms in P2P networks while keeping a good QoE and P2P efficiency. Both simulation and realistic tests show that choosing between the two approaches of Response-Delay is a trade-off: NetDel is recommended for a cost-efficient (more P2P), stable and smooth streaming, whereas BufDel is recommended when the average quality is more important than the cost-efficiency and the stability of the streaming. We plan to expand this work to further test Response-Delay with other ABR algorithms and under low latency conditions.

REFERENCES

[1] [n. d.]. Adobe HTTP Dynamic Streaming. http://www.adobe.com/products/hds-dynamic-streaming.html

[2] [n. d.]. Apple HTTP Live Streaming. https://developer.apple.com/resources/http-streaming

[3] [n. d.]. Microsoft Smooth Streaming. http://www.iis.net/downloads/microsoft/smooth-streaming

[4] A.Beben1, P.Wiśniewski, J. Mongay Batalla, and P.Krawiec. [n. d.]. ABMA+ : lightweight and efficient algorithm for HTTP adaptive streaming. In *Proceedings Int. ACM Conference on Multimedia Systems (MMSys)*.

[5] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. 2004. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)* 36 (December 2004), 335–371. Issue 4.

[6] Joachim Bruneau-Queyreix, Mathias Lacaud, Daniel Négru, Jordi Mongay Batalla, and Eugen Borcoci. 2018. Adding a New Dimension to HTTP Adaptive Streaming Through Multiple-Source Capabilities. *IEEE MultiMedia* 25 (2018), 65–78. Issue 3.

[7] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, Animesh Nandi, Antony Rowstron, and Atul Singh. 2003. SplitStream: High-Bandwidth Content Distribution in Cooperative Environments. In *Peer-to-Peer Systems II*, M. Frans Kaashoek and Ion Stoica (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 292–303.

[8] Cisco. 2019. Cisco Visual Networking Index: Forecast and Methodology, 2017-2022. *White Paper* (February 2019).

[9] Tran Thi Thu Ha, Jinsul Kim, and Jiseung Nam. 2017. Design and Deployment of Low-Delay Hybrid CDN–P2P Architecture for Live Video Streaming Over the Web. *Wireless Personal Communications* 94, 3 (01 Jun 2017), 513–525.

[10] Yang hua Chu, S.G. Rao, S. Seshan, and Hui Zhang. 2002. A case for end system multicast. *IEEE Journal on Selected Areas in Communications* 20 (October 2002), 1456 – 1471.

[11] Qi Huang, Hai Jin, and Xiaofei Liao. 2007. P2P Live Streaming with Tree-Mesh based Hybrid Overlay. *2007 International Conference on Parallel Processing Workshops (ICPPW 2007)* (September 2007).

[12] 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.

[13] Kyung-Wook Hwang, Vijay Gopalakrishnan, Rittwik Jana, Seungjoon Lee, Vishal Misra, Kadangode K Ramakrishnan, and Dan Stuart Rubenstein. 2016. Jointfamily: Adaptive bitrate video-on-demand streaming over peer-to-peer networks with realistic abandonment patterns. *Computer Networks: The International Journal of Computer and Telecommunications Networking archive* 106 (2016), 226–244.

[14] Junchen Jiang, Vyas Sekar, and Hui Zhang. [n. d.]. Improving Fairness, Efficiency and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In

IEEE/ACM Transactions on Networking (TON). 326–340.

[15] Theodoros Karagkioules, Cyril Concolato, Dimitrios Tsilimantos, and stefan Valentin. 2017. A Comparative Case Study of HTTP Adaptive Streaming Algorithms in Mobile Networks, Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video. In *Proceedings of the 27th Workshop on Network and Operating Systems Support for Digital Audio and Video*. Taipei, Taiwan, 1–6.

[16] Theodoros Karagkioules, Georgios S. Paschos, Nikolaos Liakopoulos aand Atillio Fiandrotti, Dimitrios Tsilimantos, and Marco Cagnazzo. [n. d.]. Optimizing Adaptive Video Streaming in Mobile Networks via Online Learning. *arXiv:1905.11705* ([n. d.]).

[17] Zhi Li, Xiaoqing Zhu, Josh Gahm, Rong Pan, Hao Hu, Ali C. Begen, and Dave Oran. 2014. Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale. *IEEE Journal on Selected Areas in Communications* 32 (April 2014). Issue 4.

[18] ZhiHui Lu, You Li, Jie Wu, ShiYong Zhang, and YiPing Zhong. 2008. MultiPeerCast: A Tree-Mesh-Hybrid P2P Live Streaming Scheme Design and Implementation Based on PeerCast. *2008 10th IEEE International Conference on High Performance Computing and Communications* (September 2008).

[19] Nazanin Magharei and Reza Rejaie. 2006. Understanding mesh-based peer-to-peer streaming. *ACM NOSSDAV '06*.

[20] Maria Luisa Merani and Laura Natali. 2016. Adaptive Streaming in P2P Live Video Systems: A Distributed Rate Control Approach. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 12 (2016). Issue 3.

[21] Masoud Moshref, Reza Motamedi, Hamid R. Rabiee, and Mohammad Khansari. 2008. LayeredCast - a hybrid Peer-to-Peer live layered video streaming protocol. *2008 10th IEEE International Conference on High Performance Computing and Communications* (September 2008).

[22] Ozgur Oyman and Sarabjot Singh. 2012. Quality of experience for HTTP adaptive streaming services. *IEEE Communications Magazine* 50 (April 2012), 20–27. Issue 4.

[23] Haakon Riiser, Paul Vigmostad, Carsten Griwodz, and Pål Halvorsen. 2013. Commute path bandwidth traces from 3G networks: analysis and applications. In *ACM MMsys*.

[24] Roberto Roverso, Sameh El-Ansary, and Seif Haridi. 2012. SmoothCache: HTTP-Live Streaming Goes Peer-to-Peer. In *NETWORKING 2012*, Robert Bestak, Lukas Kencl, Li Erran Li, Joerg Widmer, and Hao Yin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 29–43.

[25] Julius Rückert, Osama Abboud, Thomas Zinner, Ralf Steinmetz, and David Hausheer. 2012. Quality Adaptation in P2P Video Streaming Based on Objective QoE Metrics. In *NETWORKING 2012*, Robert Bestak, Lukas Kencl, Li Erran Li, Joerg Widmer, and Hao Yin (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–14.

[26] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. 2015. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys Tutorials* 17 (2015), 469–492.

[27] Kevin Spiteri1, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2016. BOLA: Near-optimal bitrate daptation for online videos. *IEEE INFOCOM* (April 2016).

[28] Thomas Stockhammer. 2011. Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles. In *Proceedings of the Second Annual ACM Conference on Multimedia Systems (MMSys '11)*. Association for Computing Machinery, New York, NY, USA, 133–144. https://doi.org/10.1145/1943552.1943572

[29] Truong Cong Thang, Hung Thai Le, and Anh T. Pham. 2014. An Evaluation of Bitrate Adaptation Methods for HTTP Live Streaming. *IEEE Journal on Selected Areas in Communications* (April 2014), 693–705.

[30] Hao Yin Tsinghua, Xuening Liu, Tongyu Zhan, Vyas Sekar, Feng Qiu, Chuang Lin, Hui Zhang, and Bo Li. 2009. Design and deployment of a hybrid CDN-P2P system for live video streaming: experiences with LiveSky. *MM '09 Proceedings of the 17th ACM international conference on Multimedia*, 25–34.

[31] Dongyan XuEmail, authorSunil Suresh Kulkarni, Catherine Rosenberg, and Heung-Keung Chai. 2006. Analysis of a CDN-P2P hybrid architecture for cost effective streaming media distribution. *Multimedia Systems* 11 (2006), 383–399.

[32] Xinyan Zhang, Jiangchuan Liu, Bo Li, and Y.-S.P. Yum. 2005. CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. *IEEE INFOCOM* 3 (March 2005), 2102–2111.