Divyashri Bhat, Amr Rizk, Michael Zink, Ralf Steinmetz:
Network Assisted Content Distribution for Adaptive Bitrate Video Streaming - accepted for publication.
In: ACM Multimedia Systems Conference MMSys, June 2017

Network Assisted Content Distribution for Adaptive Bitrate Video Streaming

Divyashri Bhat
University of Massachusetts
Amherst
dbhat@ecs.umass.edu

Amr Rizk
Technische Universität
Darmstadt
amr.rizk@kom.tu-darmstadt.de

Michael Zink
University of Massachusetts
Amherst
& Technische Universität
Darmstadt
zink@ecs.umass.edu

Ralf Steinmetz
Technische Universität
Darmstadt
ralf.steinmetz@kom.tu-darmstadt.de

ABSTRACT

State-of-the-art Software-Defined Wide Area Networks (SD-WANs) provide the foundation for flexible and highly resilient networking. In this work we design, implement and evaluate a novel architecture (denoted SABR) that leverages the benefits of SDN to provide *network assisted* Adaptive Bitrate Streaming. With clients retaining full control of their streaming algorithms we clearly show that by this network assistance, both the clients and the content providers benefit *significantly* in terms of QoE and content origin offloading. SABR utilizes information on available bandwidths per link and network cache contents to guide video streaming clients with the goal of improving the viewer's QoE. In addition, SABR uses SDN capabilities to dynamically program flows to optimize the utilization of CDN caches.

Backed by our study of SDN assisted streaming we discuss the change in the requirements for network-to-player APIs that enables flexible video streaming. We illustrate the difficulty of the problem and the impact of SDN-assisted streaming on QoE metrics using various well established player algorithms. We evaluate SABR together with state-of-the-art DASH quality adaptation algorithms through a series of experiments performed on a real-world, SDN-enabled testbed network with minimal modifications to an existing DASH client. Our measurements show the substantial improvement in cache hitrates in conjunction with SABR indicating a rich design space for jointly optimized SDN-assisted caching architectures for video streaming applications.

1. INTRODUCTION

The Software Defined Networking (SDN) paradigm has transformed the way networks are controlled and managed today. For example, the emergence of SD-WAN technology such as Google's B4 approach [17], has vastly improved network utilization. The separation of control and data planes allows a much more fine-grained control of network traffic than in the case of traditional networks. For example, the B4 deployment takes advantage of features of SDN such as traffic engineering and prioritization to achieve a utilizations of over 95% as compared to 40%-50% utilization of traditional networking approaches [4].

Recent works apply the software defined paradigm to resources that include *computation and storage* in addition to networks, which are coined as Software Defined Infrastructure (SDI) [31, 33]. In this paper, we present an SDI architecture that supports Adaptive BitRate (ABR) video streaming. This work augments the traditional operation of Content Delivery Networks (CDNs) by harnessing the capabilities of the SDI infrastructure. Our focus on an SDI approach for ABR video streaming is driven by the fact that video-on-demand (VoD) is the killer application in today's Internet. According to the latest Sandvine report [3], 71% of the downstream Internet traffic at peak hours in North America is real time entertainment like live streaming and video on demand. It is forecast that this will increase to 80% by 2020. Such high demand for video content requires approaches not only to efficiently transport the data but also to manage the delivery network from the content providers to the customers. Video streaming at today's scale would be unthinkable without significant infrastructure and services provided by CDNs [8, 23]. In this work, we provide an architecture that uses SDI to efficiently manage CDN networks and improve the Quality of Experience (QoE).

One characteristic that distinguishes the widespread adaptive bitrate (ABR) video streaming systems from non-ABR systems is that in case of ABR, CDN caches may not contain all quality versions of a video. Thus, providing clients with information such as the presence of qualities of desired segments in particular caches allows the client to make an educated decision on segment retrieval. Additionally, providing bottleneck bandwidth information on the paths between the client and the caches that currently host the sought segments not only aids the client's decision but also eliminates the client's need to use less accurate bandwidth estimation methods such as end-to-end probing or application layer rate estimation. This approach is denoted as *network assisted adaptive bitrate streaming*.

While the above described functionality may be partially provided in traditional, non-SDN networks, as shown for example in [14], our approach denoted (SABR - SDN assisted ABR) requires only minimal modification at the streaming

client and it reduces the load on the streaming guidance system by providing necessary information to the clients but not participating directly in the video retrieval decision made by the streaming player. This is *crucial*, since we require the clients to retain full control of their streaming algorithms for scaling and stability reasons. Hence, we design SABR with a graceful interruption property enabling the clients to ignore the network assistance information at any time. From a performance perspective, the streaming assistance system prevents significant video quality drops described, such as, in [8] by providing alternative video segment sources for different qualities. In addition to better QoE, SABR optimizes the caching architecture to improve server offloading, i.e., the percentage of client requests serviced by the caching network, as well as, the midgress, i.e., the intra-cache network traffic. These metrics are key performance indicators for content delivery networks.

In this work, we use Dynamic Adaptive Streaming over HTTP (DASH) [34] since it is a popular, open standard for ABR streaming and quite close to other ABR approaches such as Apple's HLS [2], Mircrosoft's Silverlight [5], and Adobe's HDS [1]. While we demonstrate the functionality and benefits of our approach for DASH, we believe that it can be easily extended to other ABR streaming approaches. In this paper, we make the following contributions:

- **Architecture.** We design an SDN assisted control plane architecture to support and improve ABR video streaming in CDNs.
- **Formalization.** We present a formalization of the ABR streaming problem to show the origin of the benefits of our architecture.
- **Implementation.** We implement the proposed architecture and analyze its performance. This implementation includes (*i*) an SDN measurement service and archive that is used to monitor network paths, (*ii*) SABR - a module that aggregates monitoring information and communicates with the SDN controller and the CDN caches, (*iii*) a minimally modified DASH client implementing various streaming algorithms, which makes educated decisions on segment retrieval based on the communication with SABR, (*iv*) a modified SDN controller that is used to dynamically install paths to chosen caches, and (*v*) three distinct content placement strategies that demonstrate the impact of CDN caching using SABR.
- **Analysis.** We show an extensive analysis of the SDN assisted streaming system through a series of experiments conducted in the CloudLab testbed [32]. Our results show that SDN monitoring provides better bandwidth estimates than a purely client based estimation. We show that our system not only significantly improves the QoE, e.g., the overall video quality bitrate, at the client but also reduces the server load ratio and provides higher network utilization. In addition, we investigate the impact of various content placement strategies using SABR and analyze their performance with different state-of-the-art ABR client algorithms.

Outline. The rest of this paper is structured as follows: In Sect. 2 we describe the SDI infrastructure as well as the associated monitoring and control framework. In Sect. 3, we provide the details of the REST service application and the aggregation and processing of the network monitoring information. The SDN assisted streaming client details are given in Sect. 4. We describe the evaluation environment and the extensive analysis in Sect. 5 and 6, respectively, before providing a discussion of related work in Sect. 8.

2. SOUTHBOUND: A MONITORING AND CONTROL FRAMEWORK

In this section, we describe the design of a software defined infrastructure that supports network-assisted ABR streaming through monitored network information. In particular, we present an OpenFlow (OF) [24] Southbound API that is used to orchestrate an SDN assisted CDN for adaptive bitrate streaming. Our implementation is based on the work by Adrichem et al. [37]. We use OpenFlow since it is currently the most popular instantiation of SDN, which is also enabled in the switches of a large set of vendors.

2.1 Software Defined Infrastructure (SDI)

In the following, we use the term SDI to denote a network of software defined switches that are co-located with storage and compute power. In the specific case of a CDN, this storage and compute power can be used for caching. SDIs simplify third-party policy implementation and provide network administrators with the ability to provision, monitor and efficiently control virtual networks, computation, and storage. An example of a fully operational network that offers SDI capabilities are the GENI [11] and CloudLab [32] testbeds. We use the latter for evaluation of SABR and describe this in detail in Sect. 5.1.

Fig. 1 shows the design of an SDI infrastructure to provide application services for the most popular ABR streaming instantiation, i.e., DASH. For a detailed description of DASH we refer the interested reader to [34]. Fig. 1 depicts the underlying SDI of different autonomous systems and a control plane that is tasked with flow programming and monitoring. Details on the Northbound interface and the client are given in Sect. 3 and Sect. 4, respectively.

2.2 Monitoring Infrastructure

As depicted in Fig. 1, the monitoring module is logically separated within the controller. Its task is to decide on the statistics to be monitored, as well as, the corresponding sampling times. This provides the flexibility to deploy tailored monitoring algorithms that extract only the information needed for a given application. While there has been prior work on active monitoring that demonstrates various advantages [28], we believe the performance gain of our passive system outweighs the potential benefits of an active one.

Currently, the monitoring system queries only the switches that belong to a given network path between a client and a cache or a streaming server. Only the ports that are part of these paths are monitored, which keeps the monitoring overhead to a minimum.

Given the link capacity information the system collects the bytes transferred per port over fixed intervals to determine the available bandwidth along the path, as well as, the bottleneck link. This information is used by the Northbound interface (see Sect. 3) to assist ABR streaming applications.

We logically separate the monitoring infrastructure from the controller infrastructure such that they can be deployed

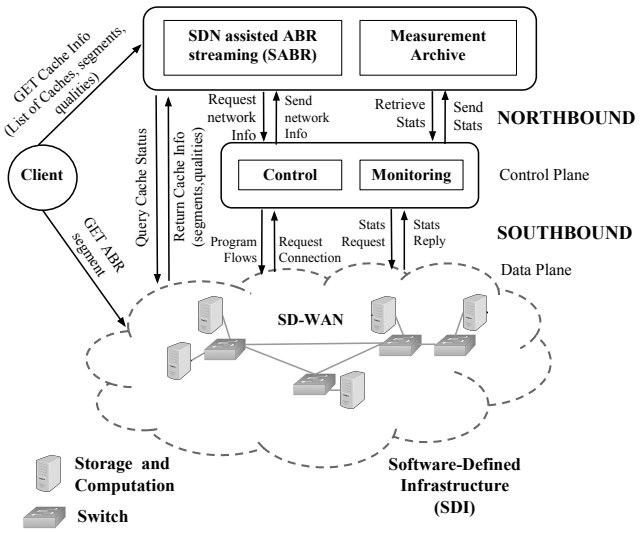


Figure 1: SDN assisted adaptive bitrate video streaming (SABR) architecture. Details of the **Southbound** interface, **Northbound** interface, **client** implementation are given in Sect. 2, Sect. 3 and Sect. 4, respectively.

on different machines. However, for optimal performance that avoids unnecessary network latency, we recommend that the monitoring infrastructure be co-located with the controller since decisions on video delivery are made based on real-time traffic. We detail the archival and processing of the monitoring information in the following section.

3. NORTHBOUND: A REST API SERVICE AND ARCHIVE

In this section, we present the Northbound interface that provides a range of information to the client, such as available caches, the bottleneck bandwidth to each cache and an indication of cache content, which can all be retrieved through standard REST APIs. Figure 1 shows the components of this system for an ABR video streaming application where the above information is sourced at an SDI as described in Sect. 2.1. In the following, we describe the system components, i.e., a REST API and monitoring archive, that use the Northbound interface.

3.1 Measurement Archive

Next, we describe the measurement archival and processing module depicted in Fig.1. OpenFlow is used to implement a feedback-based measurement and control system for an ABR video distribution system. The monitoring module from Sect. 2.2 is connected to a distributed database system powered by MongoDB [6], which enables redundancy and scalability through fast and easy replication. MongoDB also provides REST APIs for database transactions [7] enabling information insertion using simple HTTP commands.

3.2 SDN Assisted ABR Streaming - SABR

In the following, we describe the details of the SDN assisted ABR streaming (SABR) module. First, we show how SABR uses the information collected by the monitoring system to select the best cache in the network, i.e., by calculating the available bandwidth from the client to the respective caches. Further, we show how SABR provides the DASH client with monitoring information through a REST API. This information includes available bandwidth estimates and cache occupancy. This approach could easily be extended to include further switch information, e.g., queue lengths or flow table update statistics, if necessary. Finally, we use dynamic SDN routing to provide clients with the ability to connect to a desired cache. Dynamic path computation and selection within an SDN network is a generic and widely investigated optimization problem [9]. Here, we consider the specific problem of how to efficiently manage a CDN with the support of the SDN control plane. The bottleneck bandwidth is deduced from the monitored traffic information and the known link capacities. Note that in this work we do not change the routing between client and the cache pairs.

Since monitoring only gives us an observation of the past we require predictions of the future bottleneck bandwidths to help the clients determine the quality of the segments that will be retrieved next. For predicting the available bandwidth in the short term future at each port along different paths between clients and caches, we consider the well established **Auto Regressive Integrated Moving Average (ARIMA)** time series model. In the following, we denote the monitored value of transferred bytes over one port i at time slot $[t, t+\delta_i)$ as instantaneous rate $r_i(t)$. The available bandwidth is estimated based on the known link capacity and an estimate of the contending traffic.

ARIMA: The archival and processing module takes for every port i the instantaneous rates $r_i(t)$ over a history window W of time slots and provides a forecast of the utilization over the duration of N^* segments using the ARIMA time series forecasting method, which has three components, i.e., an autoregressive component of order a (AR(a)), a differencing component with parameter d and a moving average component of order v (MA(v)). In a nutshell, the differencing component removes trends by differencing d times, the autoregressive component contributes to a linear regression over the last a observed values, while the moving average components may be understood as a linear regression over the last v noise terms. We decide on the ARIMA(a,d,v) parameterization using the Akaike Information Criterion (AIC) [12], which is affine to parsimonious models that are more likely[1] to have produced the observations. In our implementation we make use of the ARIMA routines within the `R forecast` [30] library.

In the following we consider caches running variations of content placement strategies using Least Recently Used (LRU) cache eviction policy with write-through caching. Details on the parametrization of the cache algorithms and on the handling of the interaction of caches and SABR are given in the evaluation environment description in Sect. 5.3.

4. SDN ASSISTED STREAMING CLIENT

In this section, we describe an SDN assisted ABR streaming application that is based on the architecture described in Sects. 2 and 3. First, we formalize the quality adaptation problem before reviewing basic classes of adaptation algorithms which are utilized by the clients. Finally, we describe our modifications to the ABR quality adaptation algorithms and the corresponding implementation.

[1] using a maximum likelihood estimator.

4.1 The Quality Adaptation Problem

In the following, we consider a graph $G = (V, E)$ that abstracts a given network topology, where V is a set of vertices, i.e., network nodes, and $E \subseteq V \times V$ is set of links between the nodes. Each link is associated with a capacity $C(i, j)$, where i, j are the indexes of the vertices spanning the link. The network nodes are divided into three types: (i) clients, (ii) caches, and (iii) intermediate switches. For the sake of brevity, we consider a simplistic example of an ABR streaming scenario of only one video that is divided into N segments where each segment is available in K qualities, i.e., bitrate levels. Each segment carries l seconds of video while the nth segment of the kth quality has the size $X_{n,k}$ in bits. We will use k_n to denote the quality level of segment n and drop the subscript when obvious. This example can straightforwardly be expanded to the arbitrary set of available videos in the network. We assume established routing such that there exists at least one path, i.e., a set of links in E, between every client and every cache. Further, we assume that the network carries other traffic which consumes some of the bandwidth of the links, hence, $C(i, j)$ denotes the available bandwidth on the link between vertices i and j in case cross traffic is present. We assume that the available bandwidth is slowly varying with respect to the monitoring frequency of the OF switch.

Given established routing, the available bandwidth along each path between a client i and cache j is described by

$$R(i, j) = \min_{(\kappa,\iota)\in S(i,j)} C(\kappa, \iota) \ , \tag{1}$$

where $S(i, j)$ is the set of all links belonging to the path between the nodes i and j. Given the ABR streaming application, the time needed by client i to fetch the nth video segment of quality level k from cache j is given by

$$T_{n,k} = \frac{X_{n,k}}{R_{n,k}(i, j)}, \tag{2}$$

where $R_{n,k}(i, j)$ denotes the available bandwidth during the download time of segment $X_{n,k}$.

4.2 SDN Assisted Quality Adaptation

SABR provides clients with accurate in-network available bandwidth information, respectively, ARIMA-based predictions for $R_{n,k}(i, j)$. In addition, providing caching and available bandwidth information to the ABR streaming application through the Northbound interface gives the client the opportunity to minimize the fetch times $T_{n,k}$ as

$$T_{n,k} = \min_{j\in G(n,k)} \frac{X_{n,k}}{R_{n,k}(i, j)}, \tag{3}$$

where $G(n, k)$ is the set of caches, that possess the segment n in quality k. This corresponds to always choosing the cache with the highest available bandwidth. Intuitively, our approach draws its strength from the statistical multiplexing gain of combining information from independent caches such that it is less likely that the worst case conditions occur on all paths to all caches at the same time. Intuitively, the gain of the approach is stronger the more disjoint links the paths to the different caches possess.

Next, we illustrate different classes of quality adaptation algorithms showing the benefits of including SDN assistance information. Note that the modifications due to SDN assistance are orthogonal to most of the adaptation algorithms known to us such that we generally expect a performance gain, in terms of the average quality bitrate, across them.

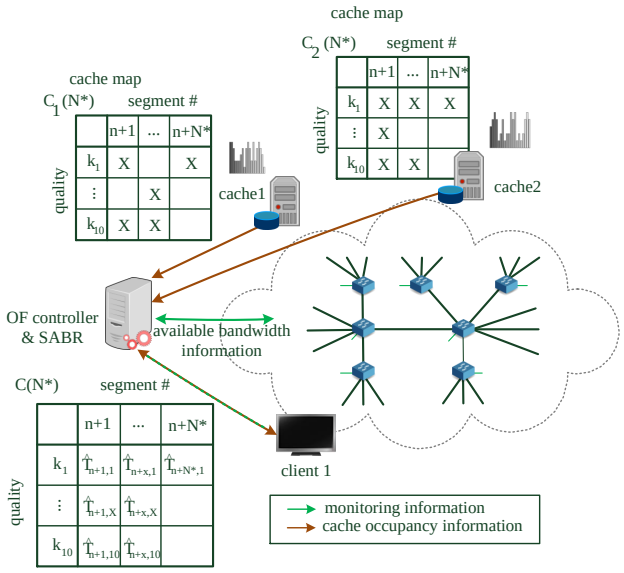


Figure 2: SDN assisted quality adaptation by providing the client with available bandwidth and cache occupancy information.

The basic quality adaptation problem can be formalized as finding the set of segments in given qualities $\{k_1, \ldots, k_N\}$ that maximize the average bitrate $\frac{1}{N}\sum_{i=1}^{N} k_i$ subject to $B(n) > 0$ for $n \in \{1, \ldots, N\}$, where $B(n)$ is the playout buffer filling after fetching segment n. Note that we measure the buffer in seconds. Stricter versions of the quality adaptation problem aim to also minimize additional QoE metrics, such as the quality variations. The quality adaptation problem is hard since clients have only smeared estimated of the available bandwidth $R_{n,k}(i, j)$ for a history of segments $\{k_1, \ldots, k_n\}$ with little to no information on the actual in-network dynamics. In the following subsection, we discuss 3 classes of quality adaptation algorithms that make use of (1) and (2) to optimize QoE before showing how SDN assistance can significantly improve their performance.

Rate-based adaptation algorithms:

Rate-based adaptation algorithms utilize estimates $\hat{R}_{n,k}(i, j)$ of (1) that are obtained at the client side to determine the quality of the next segment to be fetched. For illustration consider a very basic algorithm that would greedily download the next segment at the highest sustainable quality, i.e., the bitrate, that is just lower than the download rate of the previous segment. If the buffer is full the client idles until the end of the currently played segment. This algorithm is presented in a slightly modified fashion as VLC algorithm in [26]. Rate-based adaptation algorithms benefit from SABR as they receive available bandwidth information in the form of much more accurate ARIMA estimates of $R_{n,k}(i, j)$ in (1). These estimates outperform the empirically obtained application layer estimates at the client.

Buffer-based adaptation algorithms:

Buffer-based adaptation algorithms take only the buffer filling $B(n)$ into account when deciding on the quality of the next segment k_{n+1}. Examples for this class of algorithms include [16, 34], where the buffer space is sliced into zones that correspond to different quality adaptation behavior. SABR improves the performance of such algorithms by providing more accurate estimates for (2), i.e., the fetch time of one segment as the buffer naturally drains due to playback during this time.

Hybrid adaptation algorithms:

Hybrid adaptation algorithms take both rate and buffer information into account when deciding on the quality of the next segment (e.g., [38]). Such algorithms do not only benefit from a higher accuracy in (1) and (2), but it also utilizes (3) to find the cache with highest available bandwidth.

In general, all considered classes of quality adaptation algorithms benefit from SABR by obtaining a so-called cache map $\mathcal{C}_j(N^*)$ for every cache $j \in G$, which indicates the availability of the next N^* segments in the different quality levels at the corresponding caches. This is schematically depicted in Fig. 2. The client combines the cache maps of the different caches into a joint cache map $\mathcal{C}(N^*)$ which comprises the minimum estimated fetch times $\hat{T}_{n,k}$ for the next N^* segments in different qualities. Required estimates or lower bounds for the fetch times can be calculated from the combination of the segment sizes $X_{n,k}$ and the provided ARIMA based available bandwidth estimates for the next segments. Different quality adaptation algorithms may utilize the cache map $\mathcal{C}(N^*)$ in various ways, e.g., to optimize QoE metrics such as the average quality bitrate or the quality variation while fetching the next N^* segments. Note that SABR provides the clients with additional information, namely, $\mathcal{C}_j(N^*)$, but it *does not* control the clients' decision on which quality to fetch, which is entirely autonomous. Hence, the QoE perceived at the client fully depends on *how* the client makes use of the SABR information. We will further discuss this argument in Sect. 7. Throughout the rest of this paper we use the term Baseline to denote various non-SABR client algorithms that belong to the above classes of quality adaptation algorithms.

4.3 The Client Implementation

In order to best represent a real-world ABR streaming application, we implement our SDN assisted adaptation algorithm as part of an existing open source Python-based DASH client emulator [18]. Since each client uses HTTP by definition, we decided to let clients use a REST interface provided by MongoDB in order to minimize the implementation overhead caused by our approach. Overall, the client makes the following requests: *(i)* Initial HTTP GET request to the server or the nearest cache to retrieve Media Presentation Description (MPD) file for requested video; *(ii)* HTTP GET request to SABR for a list of qualities of next segment(s) and the available bandwidth information to every advertised cache (cf. Sect. 3.2); *(iii)* HTTP GET request to the selected cache to retrieve the desired segment.

The client parses these responses to obtain segment sizes, available bandwidth to each cache and cache occupancy information to feed it as needed to one of the algorithms described in Sect. 4.2. In order to evaluate our system we consider a miniature CDN, which is comprised of a majority of caches and a server. Details on the evaluation environment and experiment results are given in Sects. 5 & 6.

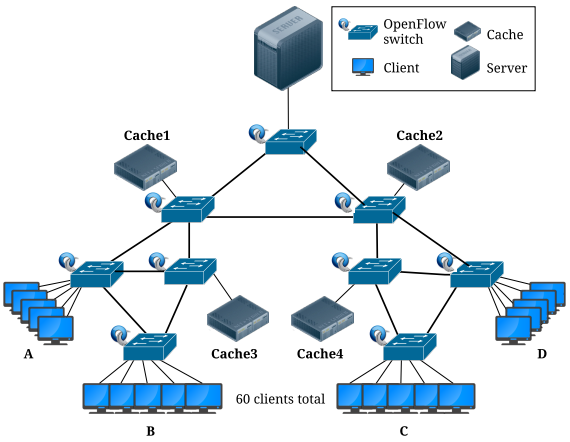


Figure 3: Cloudlab topology used for evaluating SABR. Each client group {A,B,C,D} includes 15 clients.

5. EVALUATION ENVIRONMENT

In this section, we present the experimental environment we use for the evaluation of our SDN assisted ABR video streaming approach. After introducing the Cloudlab testbed, we describe the topology and the caching algorithms used for the experiments. Evaluation results are given in Sect. 6.

5.1 Cloudlab Testbed

CloudLab [32] is a geographically distributed testbed for the development, deployment, and validation of cloud-based services. The CloudLab infrastructure consists of several different racks of varying compute and storage sizes designed to provide isolated performance and support experiments at-scale. SDN is supported through the deployment of OpenFlow switches. This highly virtualizable infrastructure is a miniature representation of SDI.

5.2 Topology

Next, we describe the topology of the Cloudlab testbed, which we use to evaluate our CDN architecture. Figure 3 shows the topology which comprises four different node types and layer-2 links of capacity 100 Mbps that connect the nodes. All nodes run Ubuntu 14.04 inside Xen virtual machines. We will identify the nodes involved in the individual experiment descriptions in Sect. 6. In the following, we describe the configuration of each node type.

Cache nodes: Cache nodes in Fig. 3 represent CDN caches serving client requests. Note that the origin server in Fig. 3 is of the same type but in contrast contains the entire video library that may be streamed in the scenario. Cache nodes run a vanilla Apache2 web server along with a HTTP packet sniffer and a MongoDB database. Together, they emulate a Web Server Gateway Interface (WSGI) that implements an LRU cache replacement policy. The Apache2 server allows persistent HTTP connections.

Client nodes: Client nodes run the different ABR algorithms we implemented in a DASH client [18] that supports SDN assisted ABR streaming as described in Sect. 4.

OVS nodes: In this topology we use software-based OF

switches that give us more flexibility in the topology generation within the testbed compared to the use of hardware OF switches. All OVS nodes run Open vSwitch 2.3.1 and communicate with a single OF controller.

OpenFlow Controller and SABR: The OF controller and SABR as depicted in Fig. 2 are installed in the same VM to minimize the REST API query, search and response time. Note that the SABR framework is logically separated from the controller and can be installed at any desired location. The associated database as described in Sect. 3.1 supports distributed implementation.

5.3 Caching algorithms with SABR

Since SABR provides a cache map $\mathcal{C}(N^*)$, which indicates the availability of the next N^* segments in different quality levels at the different caches, we **define a system-wide cache miss** as the event when the client requests a segment from the server. This event may arise either due to the absence of that particular segment at all caches or due to insufficient bandwidth to all caches. In order to analyze the performance of different caching algorithms with SABR, we implement the following content placement strategies.

Local Caching: In case of a cache miss, the system inserts this segment in the cache nearest to the requesting client. In Fig. 3 this translates to the following: *Cache1* and *Cache2* are assigned to clients in *GroupA* and *GroupC*, respectively. Similarly, *Cache3* and *Cache4* are assigned to clients in *GroupB* and *GroupD*. We use this caching strategy in all experiments described in Sect. 6.2, unless stated otherwise, to demonstrate the advantage of SABR versus a fixed cache allocation for all Baseline algorithms. This caching strategy is denoted as "Local" in Fig. 7

Global Caching: This approach assumes global knowledge of the entire cache architecture. In the first variation, denoted as "$\text{Global}_{\text{fullrep}}$", when a cache miss occurs the caching system inserts the missing segment at *all* caches. In a second variation denoted as "$\text{Global}_{\text{norep}}$", the caching system inserts the missing segment at the nearest cache only if this segment was missing in the caching system, i.e., it does not insert segments that generated cache misses due to insufficient bandwidth.

Quality-based Caching: This is a special approach as the cache space that each quality can occupy is now restricted to a subset of the global cache size. Inspired by [23], this caching system consistently maps the segments to different caches, i.e., the three lowest qualities, Q_1-Q_3, are cached in *Cache1* and *Cache2* while the higher qualities, i.e., Q_4-Q_5, are cached on *Cache3* and *Cache4*. Consistent mapping is known to perform equally good as a single contiguous cache under Zipfian and independence assumptions.

For every caching system described above, we pre-populate caches by running a set of experiments before we begin the actual measurements.

6. EVALUATION RESULTS

6.1 Evaluation Metrics and Node Setup

First, we introduce the deployed performance evaluation metrics which are partially based on metrics from [39].

Average Quality Bitrate (AQB): One of the objectives of quality adaptation algorithms is to maximize the average quality bitrate of the streamed video. For a comprehensive QoE representation, we need to combine this metric with the *Number of Quality Switches* which is explained below.

Number of Quality Switches ($\#QS$): This metric is used together with AQB and the magnitude of quality switches to draw quantitative conclusions about the perceived quality (QoE). For example, for two streaming sessions having the same AQB, the session with the lower $\#QS$ will be perceived better by the viewer.

Spectrum (H) [39]: The spectrum of a streamed video is a centralized measure for the variation of the video quality bitrate around the average bitrate. A lower H indicates a better QoE.

Rebuffering Ratio (RB): The average rebuffering ratio is given by the following equation:

$$RB = \mathsf{E}\left[\frac{t_a - t_e}{t_e}\right], \tag{4}$$

where t_a is the actual playback time and t_e is the video length in seconds, respectively.

Cache Hit Rate (C_{hr}): The cache hit rate is the average number of video segment requests that are served by the caching system, i.e., by any cache in the network divided by the overall number of requests. This ratio is usually interpreted as the probability that a video segment request is served by the caching system. It is used to assess the efficiency of the caching system.

Network Utilization (N_{util}): The average network utilization per link is given by the following formula:

$$N_{util} = \mathsf{E}\left[\frac{T_p}{C}\right], \tag{5}$$

where T_l represents the measured traffic (in bit/s) on link p and C represents the homogeneous link capacity. Note that we only measure the downstream traffic as its magnitude indicates the amount of video traffic generated in the network.

Server Load Ratio (S_{load}): The average server load ratio is the amount of video traffic (in Bytes) served from the server divided by the overall amount of video traffic received by the clients, i.e.,

$$S_{load} = \mathsf{E}\left[\frac{T_s}{T_{tot}}\right], \tag{6}$$

where T_s is the amount of traffic served by the server and T_{tot} is the total video traffic. The server load ratio is a measure for the (Byte) efficiency of the caching system.

For an extensive analysis of the SABR approach we compare the performance of three quality adaptation algorithms. We decided to compare three algorithms that map to the different categories outlined in Sect. 4.2 to better analyze the interplay between each algorithm and our network assisted approach. The algorithms are described below.

VLC [27]: This is one of the earliest DASH players which uses the following, straight-forward rate-based quality adaptation algorithm: (i) the current playout buffer filling and (ii) the average download rate of previous segments. If the buffer filling is below 25%, the client downloads the lowest quality. Otherwise, it downloads the highest quality that is sustainable based on the average download rate. If the buffer is full, the client waits for the playback duration of one segment before requesting the next one.

SQUAD [38]: This hybrid adaptation algorithm is based

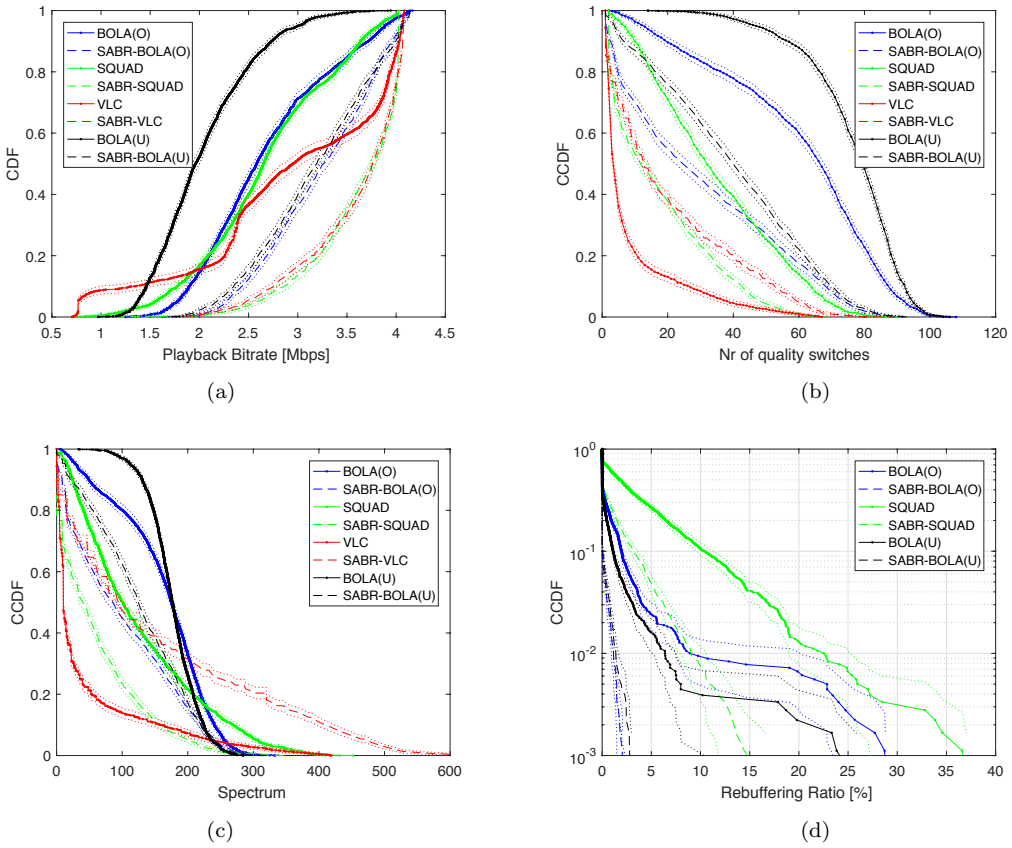


Figure 4: Main experiment: Fixed cache assignment with Baseline algorithm vs. flexible cache allocation with SABR. CCDF are given together with 0.95 confidence intervals. (i) SABR increases the playback rate (in (a)). (ii) Giving the client multiple segment sources could strongly impact the quality oscillations depending on the deployed adaptation algorithm (in (b) and (c)). Note that for (c) a lower spectrum implies less quality variations, i.e., a better QoE. (iii) SABR also significantly reduces re-buffering (in (d)).

on the spectrum metric for QoE [39]. It uses a combination of buffer and rate-based quality adaptation that accounts for the dynamics of TCP on different time scales. SQUAD possesses three states of operation: *decreasing*, *increasing* and *steady states*, that are defined in relation to a sustainable quality bitrate. SQUAD avoids sudden quality changes under fast varying available bandwidth by sacrificing buffer filling if the requested quality bitrate is sustainable.

BOLA [35]: BOLA is a buffer-based quality adaptation algorithm. It uses a Lyapunov technique for renewal processes to decide on the quality of the next segment to be fetched. While BOLA(U) aims to maximize a playback utility metric which is a weighted combination of quality bitrate and smoothness (average rebuffering time), BOLA(O) tries to minimize the oscillation in the average quality bitrate by sacrificing buffer filling without the risk of rebuffering to maintain the previously downloaded quality.

Node setup: In the following we report the results from experiments of *partial caching*, where a video might only be partially cached (in terms of segments of certain qualities at the different caches in Fig. 3). Partial caching of videos occurs when so-called *"write-through"* caching is performed, i.e., only the segments that are requested by the client in a specific quality are cached. In this scenario, the client may have to switch between caches not only because of bottleneck bandwidth fluctuations but also because of the unavailability of requested segments in a specific quality. Decisions for the latter case can be made because the SABR architecture provides the client with a cache map (cf. Sect. 4.2) that informs the client which cache possesses the desired segments at which quality levels. In the following, we report results from a series of experiments where a **Least Recently Used** (LRU) policy is implemented in all the caches.

6.2 QoE performance with SABR

In this experiment, we run 60 clients in four groups as depicted in Fig. 3. We start the clients with a time offset $w_c = 1$ sec and a client group offset of $w_g = 3$ sec. Each client streams 10 consecutive movies chosen from a set of 50 movies each of length 300 seconds where the movies are independently and identically distributed sampled from a Zipf popularity distribution with parameter $\alpha = 1$. The video qualities included in this data set are $\{89k, 0.26M, 0.79M, 2.4M, 4.2M\}bps$ and each video segment is 2 seconds long. We number the qualities Q_1 to Q_5 according to their bitrates with Q_1 being the lowest quality. We use this configuration throughout the evaluation if not stated otherwise.

In the first experiment we compare the streaming performance and the QoE metrics from Sect. 6.1 for SABR against the Baseline system of fixed client-to-cache assignments. In the Baseline system, we assign each client to the nearest cache (in number of hops) and forward cache misses to the origin server. In case of SABR we consider the caching sys-

	VLC	SABR-VLC	SQUAD	SABR-SQUAD	BOLA(U)	SABR-BOLA(U)	BOLA(O)	SABR-BOLA(O)
N_{util} (%)	42.5	71.2	44.9	74.6	42.5	68.2	46.4	65.7
S_{load} (%)	28.2	21.3	34.5	19.5	28.7	22.5	33.8	23.6

Table 1: System Performance - Network Utilization and Server Load

tem as one distributed cache as clients may request segments from different caches or the server. Note that in this experiment we only consider the *local caching* variant explained in Sect. 5.3. We specify the global cache size to be 70% of the overall dataset.

Figure 4 shows the QoE evaluation metrics for a series of 30 experiments. In Fig. 4a we clearly show that the SABR performance dominates the performance over the Baseline approach with respect to the average quality bitrate AQB for *all* evaluated quality adaptation algorithms. We observe that SABR provides a systematic gain in the streamed average quality bitrate. This gain is about 50%-100% for the clients which suffer from the lowest bitrates in the Baseline cases. We attribute this performance gain to two main factors, i.e., the flexibility in choosing the segment source and the accurate bottleneck bandwidth information that is provided to the client through network assistance. Providing bottleneck bandwidth information to the client allows it to better utilize the network as is evident from the results presented in Table 1. As shown in Table 1, using SABR reduces server load, thus, improves server offloading and potentially reduces the operational expenses of a CDN. While SABR ensures a significant gain in AQB, the improvements in the number of quality switches $\#QS$, spectrum H and the rebuffering ratio RB depend on *how* the algorithms utilize the information provided by SABR.

While the quality adaptation algorithm SQUAD aims to minimize the spectrum H, BOLA(O) focusses on reducing the oscillation in playback bitrate. Fig. 4b shows a strong reduction in quality switches $\#QS$ with the use of SABR for SQUAD, BOLA(O) and BOLA(U). Fig. 4c shows that the spectrum H of SABR variants is consistently lower than that of the Baseline algorithms. As shown in Fig. 4d, the rebuffering ratio RB is significantly lower with the use of SABR. We attribute this to the fact that SQUAD, unlike the other algorithms, uses a sliding window of segment download rate history to estimate the available bandwidth. SABR provides ARIMA-based available bandwidth predictions using a sliding window of previous available bandwidths, thus, giving a more accurate estimate of the network utilization during playback. Note that the basic VLC algorithm, which is not designed to minimize quality switches $\#QS$ or the spectrum H, shows a perceptible improvement in the overall magnitude and variance of AQB with the use of SABR, however, at the cost of increased quality oscillations. From the AQB results for VLC in Fig. 4a, we see that nearly 10% of the clients experience a low QoE and 40% of the clients are served high quality videos but with the use of SABR, the overall fairness of VLC, in terms of the spread of playback rates, is improved.

After observing significant QoE benefits with SABR for local caching, we select two distinct algorithms, BOLA(O) and SQUAD and proceed to investigate the interplay with different caching strategies as described in Sect. 5.3.

6.3 Caching performance with SABR

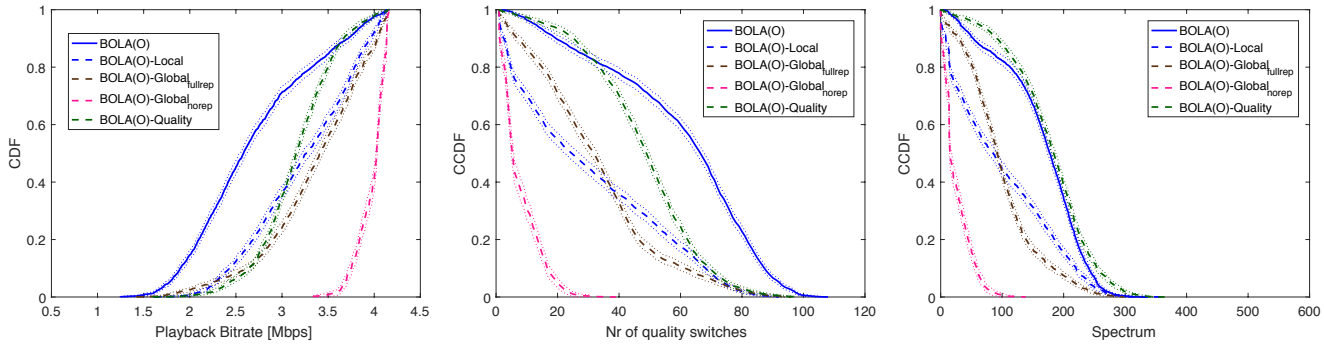
In the following, we evaluate the performance of different caching strategies as outlined in Sect. 5.3 when combined with the network assistance provided by SABR. Here, we will concentrate on two quality adaptation algorithms, i.e., BOLA and SQUAD. For the evaluation, we will resort to Fig. 5 and 6 to describe the *QoE metrics* in combination with different caching strategies for a series of 30 experiments. In addition, we present the *system performance metrics* for different caching strategies in Fig. 7 and Table 2.

Global Caching: QoE metrics in Fig. 5 indicate that BOLA(O) obtains maximum advantage from adopting a global caching strategy with no replication denoted $\text{Global}_{\text{norep}}$. The number of requests for global caching in Fig. 7a show that BOLA(O) has the highest total requests for the highest quality Q_5. The hit rates C_{hr} for other qualities are distributed almost equally. The network utilization N_{util} given in Table 2 combined with the C_{hr} in Fig. 7c demonstrates that cache hit rates increase while increasing the utilization of the cache network. Note that the utilization increases naturally as the average quality bitrate is significantly increased as can be seen from Fig. 5a.

SQUAD, on the other hand, has a high tendency to request certain qualities more than others, which is evident from the total number of requests in Fig. 7b and the cache hit rate C_{hr} in Fig. 7d, which demonstrates a higher hit rate for Q_5 than any other quality. This argument is further reinforced in Table 2 where N_{util} is the highest during full replication implying that SQUAD requests a majority of high quality segments from the caches during playback. We also observe that Fig. 6c, shows the lowest average spectrum, H, for a high replication factor, i.e., $\text{Global}_{\text{fullrep}}$ caching strategy. Note that other caching strategies such as $\text{Global}_{\text{norep}}$ and local caching achieve comparable average quality bitrate of 3.6Mbps which is reflected in the close CDFs in Fig. 6a. Combining both results we confirm that both QoE and caching efficiency are highest in SQUAD with the use of the $\text{Global}_{\text{fullrep}}$ strategy.

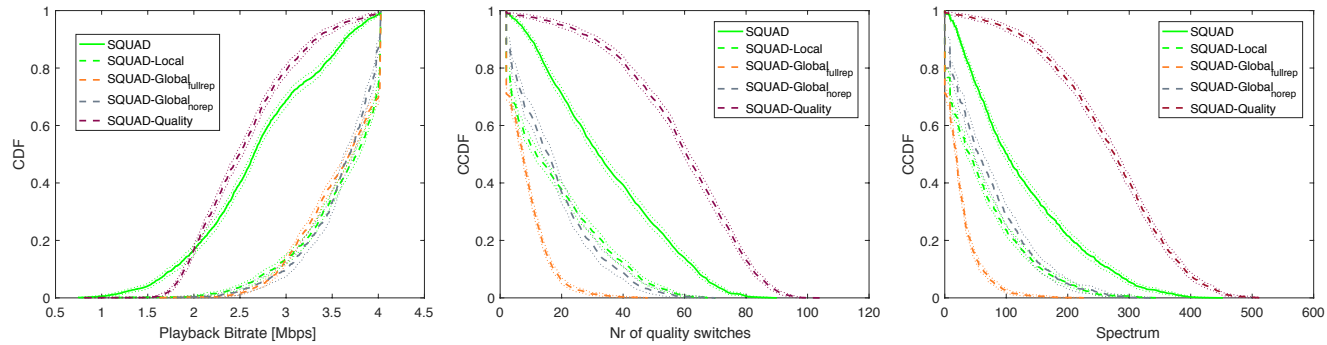
Quality-based Caching

The quality-based caching strategy studies more closely the segment quality request pattern of SQUAD and BOLA(O) clients. From Figs. 7c and 7d , we see that quality-based caching provides a consistently uniform and relatively high average hit rate C_{hr} for all qualities. However, it is important to note that while QoE for BOLA(O) is not significantly worse in this case (as seen in Fig. 5), SQUAD suffers a major loss in QoE performance, particularly noted in Fig. 6c, where this caching strategy shows the highest values for Spectrum H. Additionally, results in Table 2 for SQUAD show a comparatively low network utilization for the quality-based caching case. This information combined with the hit rate values from Fig. 7d indicate that the hit rates C_{hr} for lower qualities Q_1-Q_3 are the highest. The large number of requests for the highest quality, Q_5 in Fig. 7b and the high server load S_{load} in Table 2 leads us to infer that a majority of these high quality requests



(a) (b) (c)

Figure 5: Caching Strategies with BOLA(O) quality adaptation: QoE metrics for content placement strategies using SABR.



(a) (b) (c)

Figure 6: Caching Strategies with SQUAD quality adaptation: QoE metrics for content placement strategies using SABR.

	BOLA(O)				SQUAD			
	Local	$\text{Global}_{\text{fullrep}}$	$\text{Global}_{\text{norep}}$	Quality	Local	$\text{Global}_{\text{fullrep}}$	$\text{Global}_{\text{norep}}$	Quality
N_{util} (%)	65.7	83.3	82.9	77.1	74.6	82.3	66.9	48.2
S_{load} (%)	23.6	20.5	20.4	22.4	19.5	20.3	22.5	28.4

Table 2: Caching System Performance: Network Utilization and Server Load

are streamed from the server. We conclude that although quality-based caching improves the hit rate of low qualities in the caching system, it provides only a tolerable QoE and system performance for BOLA(O) and incurs heavy degradation in QoE metrics for SQUAD.

The evaluation we presented in this section investigates an essential trade-off between optimizing the caching architecture, specifically, the hit rates, and providing optimal QoE for the end-user. While it is evident from the results that SABR provides a vast improvement in the client streaming quality bitrate and the sever load, i.e., increasing the overall cache system hit rate, we note that a carefully selected caching strategy adopted by the CDN can also contribute significantly to the improvement of all QoE metrics.

7. DISCUSSION

7.1 Requirements of assisted streaming

Following our discussion of the results in Sect. 6 we clearly see that exploiting the information provided to the client video player by SABR or generally any network assisted streaming architecture is a non-trivial optimization problem. ABR streaming clients have been crafted in the past years to be thin and most importantly to run autonomously for scaling reasons. Network assisted streaming requires, *(i)* a well-defined API between the network and the video player, *(ii)* modifications to client algorithms to exploit this additional information, and *(iii)* an appropriate selection of caching algorithms. For example, quality adaptation algorithms such as [35, 38] depend on end-to-end bandwidth estimation methods to decide on the appropriate streaming quality. This estimate which is known to be very hard to obtain cleanly [20] can be significantly improved using network assistance. Buffer-based algorithms [16] make underlying statistical assumptions on the fetched video stream that stem from the notion of point-to-point communication which does not hold anymore if it is possible to find the *best* source for a video segment using network assistance.

It is crucial to critically assess the impact of network assistance on different streaming algorithms. On the one hand, SABR results presented in Sect. 6 show that a better streaming quality can be achieved by providing clients with

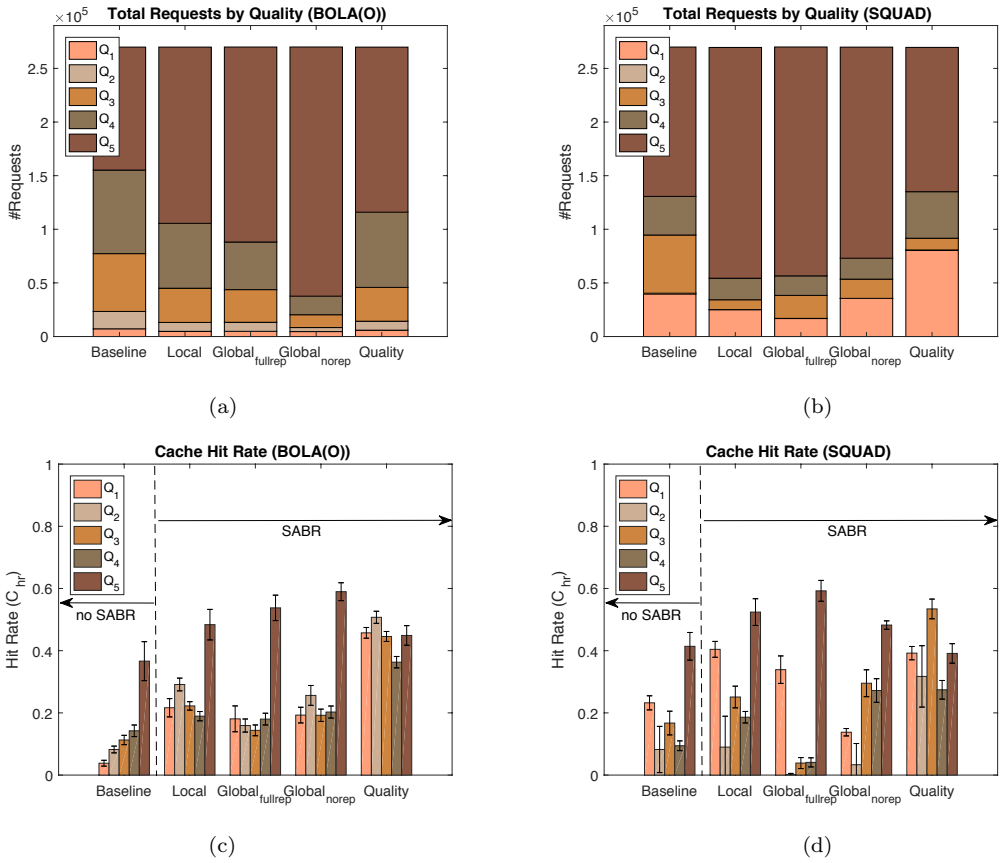


Figure 7: Absolute number of requests per quality (a-b) and cache hit rates (c-d) : Comparison of fixed content placement (Baseline) with various content placement strategies for SABR.

network information on the available bandwidth to different potential caches. On the other hand, the quality adaptation algorithm at the client may have a negative impact on QoE if it is not appropriately adapted to network assistance. For example, in case of algorithms that do not actively minimize bitrate switchings, such as `VLC`, network assistance introduces a higher number of quality switches which impairs QoE. For many other algorithms such as `SQUAD` or `BOLA` this information helps avoiding rebuffering events by providing alternative video segment sources. We also argue that SABR has the tendency to benefit some algorithms more than others. If we consider the examples of `SQUAD` and `BOLA`, both of which use a combination of rate and buffer-based approaches, `BOLA` shows higher QoE improvement with SABR. This difference can be attributed to the `SQUAD` algorithm, which uses a moving window of segment download rate history whereas `BOLA` uses the previous segment download rate to decide on the next segment quality. This implies that BOLA benefits more from the temporal information provided by the ARIMA model.

7.2 Scalability

Obviously, one concern for a system like SABR is scalability. While it is the main goal of this paper to demonstrate the applicability and benefits of SABR through evaluation in a real-world testbed, we will further investigate its performance in much larger scenarios in future work. For large-scale distributed systems such as CDNs scalability is always a concern. Therefore, we briefly discuss some of the potential scalability issues and how their impact can be diminished.

Regional Approach: Although many CDNs operate on a global scale, our approach is designed to enable operating on a regional level. Thus, several SABR systems can be used in a regional CDN instead of a global one. This will reduce the load on the system and prevent single-point of failure. In addition, our decision to use REST APIs allows a straightforward extension to a hierarchy of SABR systems, i.e., regional systems coordinated by higher-level systems.

Information on Cached Videos: As described in Sect. 4.2, SABR uses segment size and cache occupancy information, which may seem to be a huge amount of data. Luckily, only information about currently requested videos has to be retrieved. This information can be retrieved from caches using the Bloom-filter method from [23] in case of co-locating the distributed database with caches. Additionally, this information set is a much smaller subset than the set of all contents that are stored on all caches. Furthermore, with MongoDB, we have chosen a database for our system that is highly scalable and in use in large-scale systems.

Monitoring of Ports: The monitoring load of the system can be reduced by only monitoring ports that are on the paths between clients and caches. In addition, the available bandwidth predictions are centrally calculated *once* for all clients in a given prediction window.

If a SABR reply is not delivered within a certain time interval, for example, the client can always retreat to unassisted bitrate selection. Thus, our approach allows for a natural fallback into standard DASH operation at the client should SABR calls fail.

8. RELATED WORK

QoE of ABR streaming
The work by Zinner et al. [40] on application aware SDN routing looks at resource management by dynamically allocating network resources based on the status of the client buffer. The experiments in [40] look at queuing strategies for flows in the sense of the OpenFlow protocol and show that prioritization enables better traffic management, though, TCP traffic could thus suffer from short-time performance degradation. A similar approach has been explored by the authors of [19] where they provide network assistance to an ABR client for improved QoE with evaluations in a small scale wireless testbed. Cofano et al. [13] implement a Network Control Plane (NCP) to allocate and monitor a channel per video stream. In contrast, SABR shows vast improvement in QoE which we demonstrate through measurements in a large, real-world testbed without incurring the overhead of per-client QoS management. Bentaleb et. al. [10] use SDN capabilities to dynamically allocate network resources based on QoS policies to improve QoE of adaptive bitrate streaming. Their work uses a client-side probe [21] along with a DPI component to estimate the available bandwidth in a network, which leads to an overhead both for the network and control plane, respectively. The authors of [36] provide an architecture for server and network-assisted DASH denoted *SAND*. This work introduces a messaging protocol which allows QoS signalling from server to client. Similarly, in [29], Nam et al. use Network Function Virtualization (NFV) to perform MPLS traffic engineering to improve ABR streaming QoE. Our work is related in the context of computing the bottleneck bandwidth on each path. In addition, we expand on the method using time series forecasting mechanisms and by providing cache information to assist ABR quality adaptation algorithms to maximize QoE.

CDN for ABR delivery
Works that investigate the use of an OpenFlow control plane for improving video delivery in a CDN include the one by Mukerjee et al. [25], which considers the use of the control plane for load-balancing to improve QoE of live video delivery. Here, the authors design, implement and evaluate a DNS load balancing system with a hybrid (distributed and centralized) control system for live video streaming. Our work is different from this approach, firstly, in the realization through OpenAPIs via a REST interface and an OpenFlow controller, which simplifies client integration as seen in Sect. 4.3. Secondly, we consider the interplay of quality adaptation mechanisms, network bandwidth information, as well as, caching strategies.

The authors of [14] and [22] propose a coordinated control plane for routing video in the Internet. This work describes a client based monitoring and control system where the client makes intelligent decisions based on the information gathered by the monitoring system. Unlike our system, this architecture is based on a multi-CDN deployment where decisions have to be made on a client state basis using global CDN models that do not take cache occupancy into account thus contributing to considerable overhead for the control plane. In SABR, we dispense with this centralized decision model and provide assistance to ABR clients that make intelligent decisions based on the network status. In [15], Georgopoulos et al. use OpenFlow to assist a caching system, denoted OpenCache, to facilitate a client redirection to a cache based on network conditions while implementing a caching strategy of minimal hop count. Although SABR is generally able to redirect clients to caches, we additionally provide network and cache status information to the clients allowing them to make a well informed decision on content requests. We further analyze the interplay of the client-side quality adaptation and the deployed caching strategies with respect to standard QoE metrics and various performance metrics for caching systems.

9. CONCLUSIONS AND FUTURE WORK

This work leverages the potential of software defined infrastructure to provide an SDN control plane architecture that assists ABR video streaming applications in content delivery networks. This architecture, which we denote SABR, essentially, provides streaming clients with refined information on network conditions and available video sources, which is made queryable through an SDN architecture. Nonetheless, clients retain full control of the streaming decisions, including quality selection algorithms and video source switching. Our evaluation in a real-world, geographically distributed testbed shows significant improvement in quantitative metrics of QoE, which we mainly attribute to improved estimation accuracy of network characteristics, and statistical multiplexing gains. Further, we show that carefully selected caching strategies can significantly improve streaming QoE and the overall system performance.

10. REFERENCES

[1] Adobe HTTP Dynamic Streaming. http://www.adobe.com/products/hds-dynamic-streaming.html. Accessed: 2016-12-03.

[2] Apple HTTP Live Streaming. https://developer.apple.com/resources/http-streaming/. Accessed: 2016-12-03.

[3] Global Internet Phenomena Report 2016: Latin America & North America. https://www.sandvine.com/downloads/general/global-internet-phenomena/2016/global-internet-phenomena-report-latin-america\-and-north-america.pdf. Accessed:2016-12-03.

[4] Google SDN Stack. http://opennetsummit.org/archives/apr12/vahdat-wed-sdnstack.pdf. Accessed:2016-12-03.

[5] Microsoft Smooth Streaming. http://www.iis.net/downloads/microsoft/smooth-streaming. Accessed: 2016-12-03.

[6] MongoDB. https://www.mongodb.org/. Accessed: 2016-12-03.

[7] Sleepy Mongoose (MongoDB). http://www.kchodorow.com/blog/2010/02/22/sleepy-mongoose-a-mongodb-rest-interface/. Accessed: 2016-12-03.

[8] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang. Unreeling Netflix: Understanding and improving multi-CDN movie delivery. In *IEEE INFOCOM*, pages 1620–1628, 2012.

[9] S. Agarwal, M. Kodialam, and T. V. Lakshman. Traffic engineering in software defined networks. In *INFOCOM, 2013 Proceedings IEEE*, pages 2211–2219, April 2013.

[10] A. Bentaleb, A. C. Begen, and R. Zimmermann. Sdndash: Improving qoe of http adaptive streaming

using software defined networking. In *Proceedings of the 2016 ACM on Multimedia Conference*, MM '16, pages 1296–1305, New York, NY, USA, 2016. ACM.

[11] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar. GENI: A federated testbed for innovative network experiments. *Computer Networks*, 61(0):5 – 23, 2014. Special issue on Future Internet Testbeds – Part I.

[12] K. Burnham and D. Anderson. *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach*. Springer, 2003.

[13] G. Cofano, L. De Cicco, and S. Mascolo. A control architecture for massive adaptive video streaming delivery. In *ACM Workshop on Design, Quality and Deployment of Adaptive Video Streaming*, pages 7–12, 2014.

[14] A. Ganjam, F. Siddiqui, J. Zhan, X. Liu, I. Stoica, J. Jiang, V. Sekar, and H. Zhang. C3: Internet-Scale Control Plane for Video Quality Optimization. In *USENIX NSDI*, pages 131–144, 2015.

[15] P. Georgopoulos, M. Broadbent, A. Farshad, B. Plattner, and N. Race. Using software defined networking to enhance the delivery of video-on-demand. *Computer Communications*, 69:79–87, 2015.

[16] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *ACM SIGCOMM Comput. Commun. Rev.*, pages 187–198, 2014.

[17] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, J. Zolla, U. Hölzle, S. Stuart, and A. Vahdat. B4: Experience with a Globally-deployed Software Defined WAN. *ACM SIGCOMM Comput. Commun. Rev.*, 43(4):3–14, 2013.

[18] P. Juluri, V. Tamarapalli, and D. Medhi. SARA: Segment-aware Rate Adaptation Algorithm for Dynamic Adaptive Streaming over HTTP. In *IEEE ICC QoE-FI Workshop*, 2015.

[19] J. W. Kleinrouweler, S. Cabrero, and P. Cesar. Delivering stable high-quality video: an sdn architecture with dash assisting network elements. In *Proceedings of the 7th International Conference on Multimedia Systems*, page 4. ACM, 2016.

[20] R. Lübben, M. Fidler, and J. Liebeherr. Stochastic bandwidth estimation in networks with random service. *IEEE/ACM Transactions on Networking*, 22(2):484–497, April 2014.

[21] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran. Probe and adapt: Rate adaptation for http video streaming at scale. *IEEE Journal on Selected Areas in Communications*, 32(4):719–733, 2014.

[22] X. Liu, F. Dobrian, H. Milner, J. Jiang, V. Sekar, I. Stoica, and H. Zhang. A case for a coordinated internet video control plane. In *Proceedings of the ACM SIGCOMM 2012 conference on Applications, technologies, architectures, and protocols for computer communication*, pages 359–370. ACM, 2012.

[23] B. M. Maggs and R. K. Sitaraman. Algorithmic nuggets in content delivery. *ACM SIGCOMM Comput. Commun. Rev.*, 45(3):52–66, 2015.

[24] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. OpenFlow: Enabling Innovation in Campus Networks. *ACM SIGCOMM Comput. Commun. Rev.*, 38(2), Mar. 2008.

[25] M. K. Mukerjee, D. Naylor, J. Jiang, D. Han, S. Seshan, and H. Zhang. Practical, Real-time Centralized Control for CDN-based Live Video Delivery. In *ACM SIGCOMM Comput. Commun. Rev.*, pages 311–324, 2015.

[26] C. Müller and C. Timmerer. A VLC media player plugin enabling dynamic adaptive streaming over HTTP. In *ACM Multimedia*, pages 723–726, 2011.

[27] C. Müller and C. Timmerer. A VLC media player plugin enabling dynamic adaptive streaming over HTTP. In *Proc. of the ACM Conference on Multimedia*, pages 723–726. ACM, 2011.

[28] M. Mushtaq, T. Ahmed, and D.-E. Meddour. Adaptive Packet Video Streaming over P2P Networks. In *ACM International Conference on Scalable Information Systems*, InfoScale, 2006.

[29] H. Nam, K.-H. Kim, J. Y. Kim, and H. Schulzrinne. Towards QoE-aware video streaming using SDN. In *IEEE Global Communications Conference (GLOBECOM)*, pages 1317–1322, 2014.

[30] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, 2014.

[31] B. Raghavan, M. Casado, T. Koponen, S. Ratnasamy, A. Ghodsi, and S. Shenker. Software-defined Internet Architecture: Decoupling Architecture from Infrastructure. In *ACM HotNets-XI*, pages 43–48, 2012.

[32] R. Ricci, E. Eide, and The CloudLab Team. Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications. *USENIX ;login:*, 39(6), Dec. 2014.

[33] D. Simeonidou, R. Nejabati, and M. Channegowda. Software defined optical networks technology and infrastructure: Enabling software-defined optical network operations. In *Optical Fiber Communication Conference and Exposition and the National Fiber Optic Engineers Conference (OFC/NFOEC)*, pages 1–3, 2013.

[34] I. Sodagar. The MPEG-DASH Standard for Multimedia Streaming Over the Internet. *IEEE MultiMedia*, 18(4):62–67, April 2011.

[35] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *Proc. of IEEE INFOCOM*, pages 1–9, April 2016.

[36] E. Thomas, M. van Deventer, T. Stockhammer, A. C. Begen, and J. Famaey. *Enhancing MPEG dash performance via server and network assistance*. Stevenhage: IET, 2015.

[37] N. L. Van Adrichem, C. Doerr, F. Kuipers, et al. Opennetmon: Network monitoring in openflow software-defined networks. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 1–8, 2014.

[38] C. Wang, A. Rizk, and M. Zink. SQUAD: A Spectrum-based Quality Adaptation for Dynamic

Adaptive Streaming over HTTP. In *Proc. of MMSys*, pages 1:1–1:12. ACM, 2016.

[39] M. Zink, J. Schmitt, and R. Steinmetz. Layer-encoded video in scalable adaptive streaming. *IEEE Transactions on Multimedia*, 7(1):75–84, Feb 2005.

[40] T. Zinner, M. Jarschel, A. Blenk, F. Wamser, and W. Kellerer. Dynamic application-aware resource management using Software-Defined Networking: Implementation prospects and challenges. In *IEEE Network Operations and Management Symposium (NOMS)*, pages 1–6, 2014.