OSCAR: An Optimized Stall-Cautious Adaptive Bitrate Streaming Algorithm For Mobile Networks

Ahmed H. Zahran, Jason Quinlan,
Darijo Raca, Cormac J. Sreenan
Mobile and Internet System Lab.
Dept. Computer Science
University College Cork, Ireland
[a.zahran,j.quinlan,d.raca,cjs]@cs.ucc.ie

Emir Halepovic, Rakesh K Sinha,
Rittwik Jana, Vijay Gopalakrishnan
AT&T Labs – Research
1 AT&T Way,
Bedminster, NJ 08721, USA
[emir,sinha,rjana,gvijay]@research.att.com

ABSTRACT

The design of an adaptive video client for mobile users is challenged by the frequent changes in operating conditions. Such conditions present a seemingly insurmountable challenge to adaptation algorithms, which may fail to find a balance between video rate, stalls, and rate-switching. In an effort to achieve the ideal balance, we design OSCAR, a novel adaptive streaming algorithm whose adaptation decisions are optimized to avoid stalls while maintaining high video quality. Our performance evaluation, using real video and channel traces from both 3G and 4G networks, shows that OSCAR achieves the highest percentage of stall-free sessions while maintaining a high quality video in comparison to the state-of-the-art algorithms.

CCS Concepts

•**Information systems** → **Multimedia streaming;**

Keywords

HTTP adaptive video streaming, DASH, Optimization, mobile networks.

1. INTRODUCTION

Mobile data traffic has been growing at an exponential rate with the traffic volume doubling every year. Video has been a big contributor to this growth; it accounted for 55% of mobile traffic volume in 2015 and is projected to grow to 75% by 2020 [2]. Despite the increasing popularity, studies [1] in multiple countries have shown that video stalls occur in about 40-70% of all sessions. Such stalls exacerbate user frustration and ultimately lead to users abandoning their session.

It is well-known that variability in network bandwidth affects video streaming. Different techniques of Adaptive Bitrate Streaming (ABR) are used by content providers (Netflix, Hulu, YouTube). Recently, an effort to standardize ABR streaming over HTTP resulted in adoption of ISO/IEC 23009-1 standard - Dynamic Adaptive Streaming over HTTP (DASH). At a high-level, with DASH, the video is split into segments of fixed duration and encoded at multiple quality levels. Then, the appropriate quality is selected by the DASH client based on the adopted adaptation policy.

There are primarily two types of approaches for selecting video quality: rate-based [9, 6, 12] and buffer-based [8]. In rate-based, the quality of the next requested video segment depends on the client's estimate of the network throughput, whereas buffer-based algorithms make decisions solely based on their current buffer occupancy level. These heuristics, however, *do not* fully account for the variability of both the video content and wireless channel dynamics and can adversely impact the users' QoE. For example, many of the existing heuristics abstract the network throughput using its mean value independent of the nature of throughput fluctuations. This results in either maintaining low video quality by selecting lower rates to avoid stalls, or switching rates too rapidly, or in stalls while trying to maintain high video rate. After years of adaptive streaming deployment in real world, a good balance between video rate, stalls, and rate switching appears elusive.

In this paper, we present an Optimized Stall-Cautious Adaptive bitRate (OSCAR) algorithm that accommodates both video and throughput variations in a probabilistic framework. OSCAR models the throughput as a random variable and uses it to estimate a segment stall probability. Hence, OSCAR decisions are aware of variations in the underlying throughput. On selecting the next segment quality, OSCAR solves the optimization problem over a look-ahead window of several future video segments to proactively adapt to variations in video bitrate. We evaluate OSCAR through extensive ns-3 simulations that are driven by real 3G [15] and 4G traces [18] and real video content. Our results show that OSCAR achieves the highest percentage of stall-free sessions. Additionally, it attains the same quality level as the state-of-the-art buffer-based strategies [8] while reducing the number of stalls by 40%. Finally, OSCAR strikes a difficult to attain balance between high video quality and low stall performance while switching rates very smoothly.

The rest of this paper is organized as follows. Section 2 presents the background and related work. OSCAR's design is described in Section 3, followed by our performance evaluation setup and results in Section 4. Finally, we conclude and identify possible future work opportunities in Section 5.

MOVID'16, May 13 2016, Klagenfurt, Austria

DOI: http://dx.doi.org/10.1145/2910018.2910655

2. BACKGROUND AND RELATED WORK

Dynamic adaptive streaming over HTTP (DASH) represents a standardized client-server framework for media streaming [16]. DASH splits a long video into a set of smaller videos segments. These segments are encoded into different qualities and stored in HTTP servers. A media presentation description (MPD) file is usually obtained from the content provider and includes information about encoded representation rates and video resolution and the location of the segment files. There exist two types of MPD files, byte-range and segment-based [11]. In the former, each video representation is stored in a single file and segments are specified as ranges of bytes within the file. In the latter, each video segment representation is stored as an individual file with a unique URL. Once the client receives the MPD from the content provider, it usually requests the video segments in a sequential manner to fill the application buffer from which the media is then decoded and displayed. The video stalls when this buffer is empty and the application goes into a rebuffering mode to partially fill up the buffer before resuming the playout.

Different adaptation approaches are proposed for DASH. Buffer-based strategies, e.g., [8], use the buffer level as an indication for network conditions and accordingly select the next segment quality. Other algorithms estimate the network throughput by averaging the observed download rate of previous segments. Many recent algorithms [9, 6, 12, 17] employ a harmonic mean rate estimator to avoid outliers. This estimate is integrated in the adaptation logic in different ways. ELASTIC [6] employs a proportional integral controller in establishing its decision. FESTIVE [9] employs a randomized request scheduling and stateful rate-dependent switching policy in its adaptation algorithm. PANDA tries to persistently probe the link to improve its throughput estimate. MPC [17] solves an optimization problem that maximizes a user quality-of-experience metric. Harmonic mean usually provides a conservative estimate and could lead to underestimating the available bandwidth due to high variability experienced in wireless channels.

In [13], Miller et al. present an optimization framework to identify an optimal segment request strategy assuming the knowledge of long-term available bandwidth. In [4], Bokani et al. present an adaptation engine using a Markov Decision Process (MDP) based framework. The authors also propose three heuristics that use online or offline estimates for bandwidth statistics to accommodate MDP complexity. In [5], Bokani et al. integrate Q-learning to estimate the parameters of the normal distribution used to model the bandwidth. OSCAR models the throughput as a random variable and mandates a constraint on the probability on stall occurrence yielding a significant improvement in the overall performance.

3. OSCAR

OSCAR optimizes the streaming experience while accommodating variations in both video and link dynamics. To capture link dynamics, OSCAR models the network throughput as a random variable that is employed to estimate the stall probability. Additionally, OSCAR maintains a sliding look-ahead window for future video segments to capture the variations in their rates. OSCAR integrates the throughput model and actual segment information to optimize the streaming performance while maintaining a probabilistic constraint on the stall probability. In the following subsections, we first introduce our system model followed by a description of the throughput modeling and OSCAR adaptation framework.

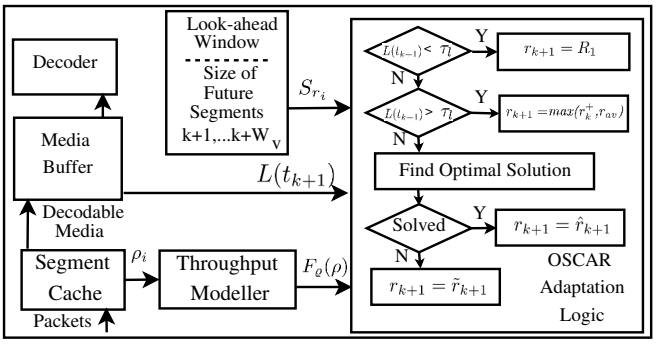


Figure 1: OSCAR Client

3.1 System Model

We consider a client streaming a video split into N segments each spanning T seconds. The streaming is assumed to be performed on a persistent HTTP connection over which segments are sequentially requested using HTTP GET requests. Each segment is encoded into Q quality representations with each representation having an average encoding rate R_q, where $q \in \{1, ..., Q\}$. Typically, higher rates imply better qualities due to a higher resolution and/or a better quantization parameter. For every segment n, the streaming client chooses the streaming rate $r_n \in \{R_1, ..., R_Q\}$ and the corresponding segment size of this segment is denoted as S_{r_n}. Generally, these segment sizes can be identified directly from a byte-range MPD file or using HTTP HEAD messages for URL-based MPD file.

3.2 Throughput Modeling

The network throughput is modeled as a random variable (RV), denoted as ϱ. The parameters of the distribution are estimated over a window of W_E throughput samples from previously received segments. Note that at the beginning of video streaming, W_E is capped to the number of received segments, denoted as k. We model ϱ using Kumaraswamy distribution [10], a sister RV to Beta distribution. Kumaraswamy enjoys the same characteristics of Beta distribution including being a double bounded distribution and ability to fit different sample distributions by tuning its two shape parameters. However, Kumaraswamy RV has the added advantage that its cumulative distribution function (CDF) can be inverted easily. The CDF of Kumaraswamy RV is expressed as

$$F_\varrho(\rho) = 1 - (1 - \rho^{\kappa_1})^{\kappa_2}, \quad (1)$$

where κ_1 and κ_2 represent the two shape parameters of Kumaraswamy distribution.

The parameters of Kumaraswamy distribution are estimated using the maximum likelihood method as described in [10]. Note that the shape parameters are estimated for normalized samples as Kumaraswamy distribution is defined over $[0, 1]$. The normalization is based on a simple linear transform of the real samples. The normalized throughput sample of segment n, denoted as $\hat{\rho}_n$, is estimated from its corresponding throughput sample, denoted as ρ_n. Note that

ρ_n is estimated as S_{r_n}/ς_n, where ς_n represents the segment fetch time for segment n and is measured starting from the time at which the segment is requested until the last byte of the segment is received. The normalized throughput is estimated using the following linear transform

$$\hat{\rho}_n = \rho_n/\rho_{max}, \tag{2}$$

where ρ_{max} represents the largest throughput sample in the considered estimation window. In order to signify the most recent samples, we employ exponentially decaying weights for the throughput samples with the most recent sample, the observed throughput for the last received segment, assigned the largest weight, denoted as ϕ_1.

3.3 OSCAR Adaptation Logic

OSCAR represents a stall-cautious hybrid client that combines both buffer-based and rate-based strategies. Similar to buffer-based clients, the buffer is divided into three regions including low, transient, and high. Let t_{k+1} represent the request time of next segment, the first segment in the look-ahead window, and $L(t_{k+1})$ represent the buffer level in seconds at t_{k+1}. At low buffer levels, identified by $L(t_{k+1})$ below τ_l seconds of video, the client requests the video at the lowest quality to ramp up the buffer level before it starts optimizing the video quality. At the high buffer level, identified by $L(t_{k+1})$ higher than τ_h, the client targets streaming high quality video while avoiding OFF periods to maintain an accurate estimate of the available throughput. To achieve these goals, the client requests the next segment at the quality rate corresponding to the maximum of the rate of the representation that is just higher than the last requested rate, denoted as r_k^+, and the representation whose rate is just below the average throughput, denoted as r_{av}. Note that going to OFF state is unavoidable if the link is good, as the buffer will quickly fill up. Typically, streaming clients remain OFF until the buffer can accommodate at least one more segment, i.e., the client starts downloading the new segment once it completes the playout of the current head of line video segment.

In the transient region, the client optimizes the streaming decision using the optimization model described in the following subsection. The solution of the optimization problem provides the optimal rate for the look ahead window consisting of few future segments. The client then requests the next segment at its estimated optimal rate, denoted as $\hat{r}_{k+1}$. However, if the problem is infeasible, the next segment is requested at a fallback rate, denoted as $\tilde{r}_{k+1}$, that is determined according to the following policy. A rate bound, denoted as r_b, is identified as the minimum rate in the throughput estimation window. The client requests the highest quality whose rate is below r_b and is at most n_b switching levels away from the last received segment rate, i.e., r_k. This level switching bound is designed to avoid abrupt quality changes when an optimal rate cannot be determined. Figure 1 highlight OSCAR client operation.

3.4 OSCAR Optimization Model

We define our objective as a weighted sum of the selected video utility and switching penalty over a look-ahead window including the next W_V segments. Concave video utility functions, such as exponential and log, are commonly used in the literature. In this work, we consider an exponential video utility, denoted as $U(r_n)$, that is expressed as

$$U(r_n) = 1 - exp(-r_n/(R_Q\overline{r})), \tag{3}$$

where $\overline{r}$ is a parameter that represents the device capability and R_Q is the highest encoding rate.

We also consider a quality switching penalty that favors incremental quality switches to large quality switches. Our quality switching penalty for segment n, denoted as $P(r_n)$, is expressed as

$$P(r_n) = \left(\frac{r_n - r_{n-1}}{R_Q}\right)^2. \tag{4}$$

This function would favor performing two small quality changes instead of one large quality change to smooth out quality transitions. Hence, it will lead to more smooth switches. Additionally, we take advantage of the proposed look-ahead window to improve switching dynamics by introducing two constraints: an oscillation avoidance constraint and a stall avoidance constraint. The former constraint targets smoothing out possible temporary changes in the selected quality due to video bit rate variations across segments. To illustrate, small segments may trigger unnecessary quality up switches that are followed by down switches leading to a noticeable quality variation. Hence, oscillation avoidance is attained by enforcing that the selected quality rates in the look-ahead window should be monotonic and is expressed as

$$\sum_{n=k+1}^{k+W_V} |r_n - r_{n-1}| = |r_{k+W_V} - r_k|. \tag{5}$$

At any decision instant, the size of the different segment representations for the next W_V segments is considered in the decision. Note that W_V is capped to the number of the remaining segments towards the end of the video.

Stall avoidance constraint is another core constraint in OSCAR that mandates a probabilistic guarantee on stall avoidance in the look-ahead window. In order to avoid stalls, the arrival time of a segment n, denoted as a_n, should be before its playout time, denoted as D_n. Hence, the stall avoidance constraint is defined as imposing a threshold, denoted as γ, on the probability of the segment arrival before its deadline; i.e. $P(a_n < D_n) > \gamma$. The arrival time for any segment n in the look-ahead window is estimated as the time required to transmit all segments up to and including segment n and is expressed as

$$a_n = \sum_{i=k+1}^{n} S_{r_i}/\varrho \ \ \forall n = k+1, ..., k+W_V. \tag{6}$$

In OSCAR, the deadline of the first segment (D_{k+1}) in the look-ahead is conservatively determined as the buffer level less the duration of two segments; i.e. $D_{k+1} = L(t_{k+1}) - 2T$. The deadline of subsequent segments is incremented by one segment duration T for each subsequent segment in the look-ahead window; i.e., $D_n = D_{k+1} + T * (n - k - 1)$. Hence, the stall avoidance constraint for any segment n within the look-ahead window can be expressed as

$$P(\sum_{i=k+1}^{n} S_{r_i}/\varrho < D_n) > \gamma \ \ \forall n = k+1, ..., k+W_V, \tag{7}$$

which can also be rewritten as

$$\sum_{i=k+1}^{n} S_{r_i}/D_n < F_\varrho^{-1}(1-\gamma) \ \ \forall n = k+1, ..., k+W_V. \tag{8}$$

where $F_\varrho(\cdot)$ is the CDF of the throughput.

Hence, we formally define the OSCAR optimization problem as

$$\max_{r_n} \sum_{n=k+1}^{k+W_V} U(r_n) - \alpha P(r_n) \quad (9)$$

$$\text{s.t.} \sum_{i=k+1}^{n} S_{r_n}/D_n < F_\varrho^{-1}(1\text{-}\gamma)\ \forall \text{n=k+1,...,k+}W_V \quad (10)$$

$$\sum_{n=k+1}^{k+W_V} |\ r_n - r_{n-1}\ | = |r_{k+N} - r_k| \quad (11)$$

$$r_n \in \{R_1, ..., R_Q\}\ \forall n \in \{k+1, .., k+W_V\}$$

where α represents a weighting factor that can be tuned to adjust the relative weight of the switching penalty to the video utility. Constraint (11) can linearized using absolute value linearization techniques. The details of this linearization is removed due to space limitation.

The resultant problem represents a mixed nonlinear integer program that can be solved to identify the optimal streaming rates for the following look ahead window. However, a solution is quickly obtained as the maximized objective function is concave and the constraints are all linear in the program variables. The program is implemented as a multiple choice knapsack problem and can be solved using any suitable solver. In this work, we used LINDO solver[1].

To this end, one can tune the algorithm performance by changing the stall probability threshold γ (in Eqn 10) to match different user profiles. A larger value of γ implies a more stall conservative user, which is usually attained by reducing the received video quality. Additionally, α (in Eqn 9) can be tuned to adjust switching relative weight to perceived quality. It is possible that a content provider would set these parameters by monitoring individual user reaction to the delivered content [3].

4. PERFORMANCE EVALUATION

In this section, we first present our evaluation setup followed by our performance metrics and results.

4.1 Evaluation Setup

The performance of OSCAR is evaluated using ns-3 simulator. Our evaluation setup is shown in Figure 2. We implemented two modules for the DASH client and server that communicate over a single TCP connection. The communication link bandwidth is shaped according to throughput traces for both 3G (54 traces) [15] and 4G/LTE (23 traces) [18][2]. These traces are collected while riding different modes of transportation [15, 18]. Each trace reports the measured network throughput over subsequent observation periods with each period spanning approximately one second.

In our simulations, we used six five-minute H.264 clips including Big Buck Bunny (Animation), Sita Sings the Blues (Animation), and Elephant Dreams (Animation), Clip_4 (Animation), Clip_12 (Documentary), Clip_16 (Action) from the Dash dataset [14][3], where videos are encoded with 4-second segment at {(235, 320x240), (375, 384x288), (560, 512x384), (750, 512x384), (1050, 640x480), (1750, 720x480), (2350, 1280x720), (3000, 1280x720), (3850, 1920x1080), (4300, 1920x1080)}.

[1]http://www.lindo.com/

[2]We modified the implementation of ns-3's point-to-point link to disable transmissions when the traces have zero throughput.

[3]www.cs.ucc.ie/misl/research/current/ivid_dataset

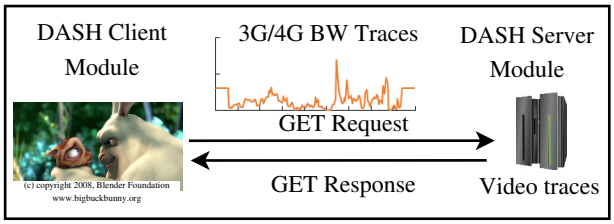


Figure 2: Evaluation Setup

Table 1: The parameters of streaming algorithms

BBA [8]*	$\underline{r} = 2T$	$\bar{r} = 0.6B$	$\Delta B = 0.875T$
	$\tau_h = 0.9B$		
ELASTIC [6]*	$W_E = 5$	$k_p = 0.01$	$k_i = 0.001$
MPC [17]*	$\lambda = 1$	$\mu = 8600$	$\mu_s = 8600$
	$W_E = 5$	$h = 5$	
OSCAR	$W_E = 10$	$W_V = 4$	$\tau_l = 12s$
	$n_b = 3$	$\phi = 0.4$	$\tau_h = 54s$
	$\alpha = 1$	$\gamma = 0.999$	

* Default parameter values are used from the cited papers

The DASH client requests video segments using HTTP GET request and the DASH server replies back with a file whose size equals that identified by the output of the aforementioned encoding stage. The requests are instantaneously sent in a sequential manner upon receiving the previous video segment unless the buffer is full. If the buffer is full, an algorithm dependent request delay is introduced. If the algorithm does not have a request scheduling policy, the next segment is requested once the buffer can accommodate a new segment. In our implementation, we consider a fixed HTTP header size of 100 bytes for both requests and responses. The default values of the DASH client include a 60s buffer, 8s of initial buffering, and 4s of rebuffering before resuming playout.

We compare the performance of OSCAR to the following streaming algorithms; ELASTIC [6] represents the class of rate-based algorithms. W_B is the sample size for rate estimation, while k_p and k_i are the proportional and integral coefficients of the controller. BBA [8] represents the class of buffer-based algorithms. In our evaluation, we implemented BBA-2 variant with linear mapping between the reservoir and the upper buffer threshold. $\bar{r}$ and $\underline{r}$ represent the upper and lower bounds of the reservoir, τ_h is the upper reservoir threshold, and ΔB is the the quality improvement metric defined in [8]. MPC [17] represents the class of optimized streaming algorithms. MPC program is implemented using the Lindo Solver[4]. W and N are the throughput averaging and look-ahead window, respectively, while λ, μ, and μ_s represent the weights of the objective function. As suggested in [17], we set $\mu = \mu_s$ to double the maximum encoding rate to signify the stall component for a fair comparison. Table 1 shows the values of the parameters of different algorithms.

Our performance metrics include the average received qual-

[4]We implemented Robust MPC that considers a conservative estimate of the mean estimated network available bandwidth

Table 2: Average Performance Metrics

Algorithm	n_{st}	t_{st}	r_{av}	n_{sw}	l_{sw}	ζ
BBA	0.95	5.59	1467	24.74	1.71	0.64
ELASTIC	0.47	2.25	935	13.21	1.24	0.44
MPC	2.30	14.16	1699	22.93	2.07	0.68
OSCAR	0.56	4.33	1461	27.7	1.67	0.65

ity rate (r_{av}), the average number of stalls (n_{st}), the average stall duration (t_{st}), the average number of switches (n_{sw}), the average switching level (l_{sw}), and the average network bandwidth utilization (ζ). These metrics are averaged per session. In this context, the average bandwidth utilization represents the percentage of bandwidth used over the session lifetime and is estimated as the ratio of the transmitted data to the maximum amount of data that could have been transmitted for a given trace during the session activity.

4.2 Simulation Results

Table 2 shows the averaged performance metrics over 522 (87 traces x 6 videos) trace-video pairs. The table shows that ELASTIC achieves the lowest average number of stalls per session, n_{st}, and average total stall duration per session, t_{st}. This comes at a very high cost of reduced average video rate by 1.6x compared to OSCAR and BBA. ELASTIC performance is attributed to its conservative bandwidth estimate, leading to rapid filling of the video buffer. Hence, ELASTIC stalls for a shorter duration when the link goes down for a longer period and can avoid medium-term link degradation. BBA incurs approximately twice the number of stalls as ELASTIC with more than 2.5x stall duration.

OSCAR strikes a good balance between achieving high video quality with a significant reduction in the number of stalls encountered by BBA using agile rate switching, n_{sw} and l_{sw}. This leads to significantly better stall results (about 40% fewer stalls than BBA), while maintaining the same average quality as BBA. OSCAR's stall results are not quite as good as those of ELASTIC, but close analysis reveals that these additional stalls are limited to only 3% of the traces where conditions are extremely challenging, i.e. the bandwidth sharply drops for extended durations for which stalls can be only avoided by relying on buffered media. It is worth noting that OSCAR attains an average streaming rate that is 1.6x that attained by ELASTIC. The table also shows that OSCAR performs more switches than others. However, these switches are smooth and stay near the average level of switching of other algorithms. In [7], Egger et. al. show that frequent switching did not degrade the user QoE for segment sizes 4-10 seconds, which are typically used in mobile networks. OSCAR also attains the second highest bandwidth utilization, ζ, after MPC while ELASTIC shows very low bandwidth utilization due to its conservative strategies.

Figure 3 plots the cumulative distribution function (CDF) of different metrics for the simulated algorithms. Figure 3a shows that OSCAR and BBA choose very similar rates while MPC tends to choose higher rates and ELASTIC tends to choose lower rates. OSCAR and BBA are more efficient than ELASTIC in using the available bandwidth (Figure 3b). Figure 3e shows a noticeable variation in the distribution of number of rate switches for different algorithms. Interestingly, MPC and ELASTIC did not change their rate in 5% of the simulated sessions. The figure shows that ELASTIC performs the least number of switches as its CDF ramps up earlier. OSCAR performs a higher number of rate switches as it adapts to link and video variations. Figure 3f illustrates that the switches for OSCAR are smooth, with the average number of levels switches remaining well below 2. Figure 3c shows that OSCAR attains the highest percentage of stall free sessions at 85.4% followed by ELASTIC at 83.33% and then BBA at 66.3%. Figure 3d shows that ELASTIC enjoys the shorter stall duration followed by OSCAR then BBA and finally MPC.

Table 3: Performance for high variability traces.

Algorithm	n_{st}	t_{st}	r_{av}	n_{sw}	l_{sw}	ζ
BBA	1.07	7.136	1419	24.89	1.73	0.69
ELASTIC	0.47	2.62	851	13.84	1.24	0.46
MPC	2.74	18.20	1656	23.81	1.83	0.73
OSCAR	0.61	5.54	1398	29.36	1.66	0.69

Table 4: Performance for low variability traces.

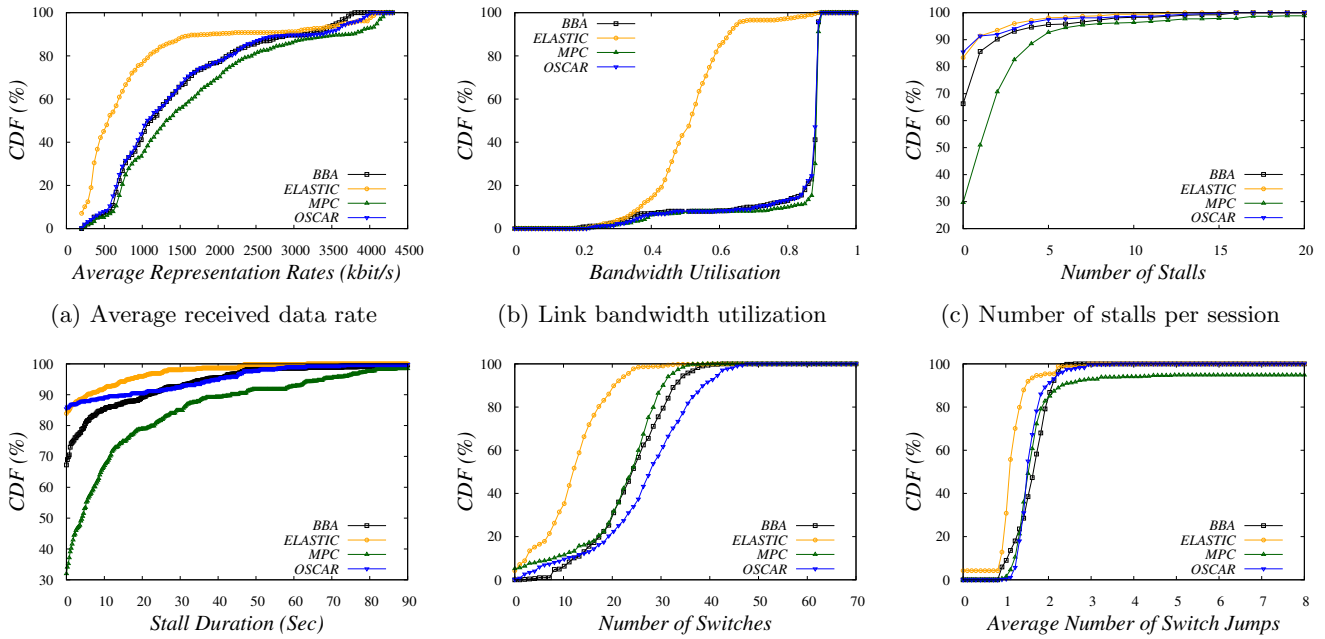
Algorithm	n_{st}	t_{st}	r_{av}	n_{sw}	l_{sw}	ζ
BBA	0.63	1.77	1588	24.37	1.66	0.55
ELASTIC	0.46	1.35	1145	11.65	1.23	0.42
MPC	1.34	4.46	1814	20.00	2.67	0.59
OSCAR	0.44	1.35	1620	23.7	1.71	0.58

4.3 Impact of bandwidth variability

We next consider the impact of bandwidth variability on the performance of adaptation strategies. To this end, we divide all traces into two profiles according to the Coefficient of Variation (CoV) of observed bandwidth: (i) low variability profile, where $CoV < 0.5$ and (ii) high variability, where $CoV \geq 0.5$. We end up with 150 traces with low and 372 traces with high variability. Table 3 shows the results for high throughput variability traces. ELASTIC results in 23% reduction in stalls in comparison to OSCAR but this reduction is accompanied by 40% reduction in the achievable average rate. Compared to BBA, OSCAR has a similar average rate with only 56% of BBA stalls and 78% of its average stall duration. Table 4 shows the detailed results for low variability traces. For these traces, OSCAR produces the least number of stalls and average stall duration. MPC attained the best average quality rate that is approximately 11% higher than OSCAR's average rate but with 3x the number of OSCAR's stalls. Both BBA and OSCAR attain the same average quality rate that is 1.4x that of ELASTIC.

5. CONCLUSION

In cellular networks, video clients are required to accommodate the variability in both link and video dynamics that significantly affects the streaming performance. We presented OSCAR, a novel streaming algorithm that optimizes adaptation decisions to improve video quality while reducing stall probability and providing smooth quality switches. Our performance evaluation, using real video files and channel traces from operational cellular networks, shows that OSCAR achieves the largest percentage of stall-free sessions in comparison to state-of-the-art algorithms. Additionally, we show that OSCAR strikes a balance between streamed video quality and stall performance in comparison to state-of-the-art algorithms. For future work, we are developing adaptation heuristics based on the OSCAR optimization frame-



(a) Average received data rate (b) Link bandwidth utilization (c) Number of stalls per session

(d) Total stall duration per session (e) Number of quality switches per session (f) Average switching level per session

Figure 3: CDF of performance metrics

work to enable a lightweight implementation for streaming clients.

6. ACKNOWLEDGMENT

This publication has emanated from research conducted with the financial support of Science Foundation Ireland (SFI) under Grant Number 13/IA/1892.

7. REFERENCES

[1] "Mobile operators and their customers still struggling with video stalling, new Opera study finds. Available at: http://goo.gl/exXrO1."

[2] "Cisco Visual Networking Index Global Mobile Data Traffic Forecast Update 2015-2020. Available at: http://goo.gl/jFB2L7," 2015.

[3] A. Balachandran et al., "Developing a Predictive Model of Quality of Experience for Internet Video," in *Proc. of the ACM SIGCOMM*, August 2013, pp. 339–350.

[4] A. Bokani et al., "HTTP-Based Adaptive Streaming for Mobile Clients using Markov Decision Process," in *Proc. IEEE PV*, Dec 2013.

[5] ——, "Optimizing HTTP-Based Adaptive Streaming in Vehicular Environment Using Markov Decision Process," *IEEE Trans. on Multimedia*, vol. 17, no. 12, pp. 2297–2309, Dec 2015.

[6] L. De Cicco et al., "ELASTIC: A Client-Side Controller for Dynamic Adaptive Streaming over HTTP (DASH)," in *Proc. of IEEE PV*, Dec 2013.

[7] S. Egger et al., "The Impact of Adaptation Strategies on Perceived Quality of HTTP Adaptive Streaming," in *Proc. of VideoNext*, Dec. 2014, pp. 31–36.

[8] T.-Y. Huang et al., "A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service," in *Proc. of ACM SIGCOMM*, Aug. 2014, pp. 187–198.

[9] J. Jiang et al., "Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE," in *Proc. of CoNEXT*, Dec. 2012, pp. 97–108.

[10] M. Jones, "Kumaraswamyś distribution A beta-type distribution with some tractability advantages," *Statistical Methodology*, vol. 6, no. 1, pp. 70 – 81, 2009.

[11] V. Krishnamoorthi et al., "Helping Hand or Hidden Hurdle: Proxy-Assisted HTTP-Based Adaptive Streaming Performance," in *Proc. IEEE MASCOTS*, Aug 2013, pp. 182–191.

[12] Z. Li et al., "Probe and Adapt: Rate Adaptation for HTTP Video Streaming At Scale," *IEEE J. Sel. Areas in Commun.*, vol. 32, no. 4, pp. 719–733, 2014.

[13] K. Miller et al., "Optimal Adaptation Trajectories for Block-Request Adaptive Video Streaming," in *Proc. of IEEE PV*, Dec 2013, pp. 1–8.

[14] J. J. Quinlan et al, "Datasets for AVC (H.264) and HEVC (H.265) for Evaluating Dynamic Adaptive Streaming over HTTP (DASH)," in *Proc. of ACM MMsys 2016 (dataset track) (to appear)*, May 2016.

[15] H. Riiser et al., "Commute Path Bandwidth Traces from 3G Networks: Analysis and Applications," in *Proc. of ACM MMSys*, Feb 2013, pp. 114–118.

[16] T. Stockhammer, "Dynamic Adaptive Streaming over HTTP –: Standards and Design Principles," in *Proc. of ACM MMSys*, 2011, pp. 133–144.

[17] X. Yin et al., "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proc. of SIGCOMM*, Aug 2015, pp. 325–338.

[18] X. K. Zou et al., "Can Accurate Predictions Improve Video Streaming in Cellular Networks?" in *Proc. of HotMobile*, Feb 2015, pp. 57–62.