HTTP-based Adaptive Streaming for Mobile Clients using Markov Decision Process

Ayub Bokani, Mahbub Hassan, Salil Kanhere
School of Computer Science and Engineering, The University of New South Wales, Sydney 2052 NSW, Australia
Email: {abokani, mahbub, salilk}@cse.unsw.edu.au
National ICT Australia (NICTA), Alexandria 1435 NSW, Australia
Email: {Ayub.Bokani, Mahbub.Hassan, Salil.Kanhere}@nicta.com.au

***Abstract*—Due to its simplicity at the server side, HTTP-based adaptive streaming has become a popular choice for streaming on-line contents to a wide range of user devices. In HTTP-based streaming systems, the server simply stores the video segmented into a series of small chunks coded in many different qualities and sizes, and leaves the decision of which chunk to download next to achieve a high quality viewing experience to the client. This decision making is a challenging task, especially in mobile environment due to unexpected changes in network bandwidth as the user moves through different regions. In this paper, we consider Markov Decision Process (MDP) to derive the optimum chunk selection strategy that maximizes streaming quality and propose three approaches to reduce the computational cost of MDP. The first approach recomputes the solution after downloading every k chunks. The second approach computes the solution once using global network statistics of a given region. The third approach recomputes the solution every x meters using offline statistics for each x meters of the road. The three approaches are compared using real-world 3G bandwidth and mobility traces. The best performance is achieved with x-MDP.**

I. Introduction

Due to immense scalability benefits, there is a strong push from the industry to adopt HTTP as a universal platform for delivering all types of contents, including video. Apple [1], Microsoft [2], and Adobe [3] are already trialling their own proprietary HTTP-based video streaming platforms while a standard, called dynamic adaptive streaming over HTTP (DASH) [4], has been recently drafted by the world wide web consortium (W3C) to facilitate wide-spread deployment of this technology.

The key concept in DASH is to code the same video in multiple bitrates (qualities) and store each stream into a series of small video chunks of 2-4 sec durations. A client simply downloads and plays a chunk of a given quality using the standard HTTP GET command used for fetching any other objects on the Web. Since video has strict display deadlines for every frame, each chunk needs to be downloaded before its deadline to avoid the 'freezing' effect. It therefore becomes the responsibility of the client to dynamically select the 'right' quality of the next chunk to ensure a smooth video at the receiver with the highest possible quality and minimum number of quality switches from one chunk to the next. The DASH standard only specifies how the video chunks should be stored and what metadata about the chunks should be provided to a client. The actual *streaming strategy*, i.e., the client intelligence for selecting the right quality for each chunk in order to produce a high quality of experience (QoE) for the viewer is left to the developers.

There is a growing evidence in the recent literature [5]–[8] that the streaming strategy under uncertain network bandwidth can be effectively optimized using Markov Decision Process (MDP). MDP is an optimization framework that maximizes a revenue function with arbitrary weights assigned to quality, deadline miss, as well as quality change frequency. By adjusting these weights, a client can optimize QoE with the desired balance over these three video quality dimensions. Because MDP is based on solving an optimization problem, it is expected to provide superior performance over heuristic-based rate adaptation algorithms. Indeed, researchers have confirmed that MDP outperforms many existing heuristic adaptations in various networking scenarios [6], [8].

A major issue with MDP-based streaming adaptation is the high computation cost of solving the complex optimization problem, especially if it needs to be re-solved online many times during the same streaming session. For highly mobile clients, such as those located inside a moving car, it may be difficult to model the network bandwidth distribution with a fixed set of parameter values as bandwidth distribution is often found to be influenced by the location [9]. If the bandwidth model parameters change, the optimum solution does not remain optimum anymore unless the problem is re-solved using the new parameter values. Thus, for the best MDP performance in a highly mobile environment, we are confronted with the task of deciding when (or how frequently) the MDP optimization is to be solved, which directly affects the streaming performance as well as the MDP computation overhead. To the best of our knowledge, previous work has not addressed this particular MDP issue.

In this paper, we propose and evaluate three approaches to reduce MDP overhead. We consider both on-line and offline MDP optimization. For online, we propose a k-chunk update approach (k-MDP) which recomputes the optimal strategy after downloading every k chunks. The online approach uses the bandwidth statistics gathered only during a video session without referring to any statistics collected offline. For offline, we propose two different approaches, single MDP strategy (s-MDP) and x-meter multiple MDP strategy (x-MDP). s-MDP uses the global statistics of a given region to compute an optimal MDP strategy which is used throughout the video session in any given trip in that region, while x-MDP recomputes the optimal strategy for every x meters of the travel using offline

978-1-4799-2172-0/13/$31.00 ©2013 IEEE

statistics for the last x-meter of the road.

We have evaluated the performance of the proposed approaches using simulation driven by real-world 3G bandwidth and vehicular mobility traces. We find that k-MDP is capable of reducing online computation overhead significantly without any noticeable degradation in video QoE. Interestingly, although offline approaches have zero online computation overhead, they both outperform the online approach. The best performance is achieved with x-MDP.

The rest of the paper is organised as follows. We discuss related work in Section II. Section III shows how DASH can be formulated as an MDP problem. We present the proposed MDP overhead reduction approaches in Section IV, followed by the simulation details in Section V. Results are presented in Section VI. We conclude the paper in Section VII.

II. Related work

Yao et al. [10] and Deshpande et al. [11] have collected bandwidth traces from 3G networks while driving in a car and using an entropy-based method have shown that road-segment-based statistics contain more information about bandwidth compared to global statistics. Authors of [12], [13] have successfully implemented and tested a platform that allows streaming clients to access bandwidth statistics history of a given location in real-time, giving evidence that location-based streaming optimization is technically viable. Yao et al. [9], Halvorsen et al. [12], Curcio et al [14], and Singh et al. [13] have investigated the use of location-based bandwidth statistics to improve video streaming performance, but they have not considered the MDP optimization framework.

There is a growing body of literature on the use of MDP to optimize video streaming. Cuetos and Ross [15] used MDP to jointly optimise scheduling and error concealment in layered video, while Mastronarde et al. [16] have recently demonstrated that the power consumption problem of video decoding can be effectively modelled as an MDP. In [17], the authors used MDP to model the prefetching decision problem in adaptive multimedia. Xing et al. [8] and Xiang et al. [7] have considered MDP to optimize rate adaption of streaming video where the uncertainty in network bandwidth is modelled as a Markov chain with its own bandwidth states. The MDP formulation we used to model the details of DASH is closest to the ones reported in [5], [6]. However, the author of [5], [6] have not considered the issue of MDP complexity overhead. Proposing and comparing different strategies to reduce MDP computation cost in highly mobile environments is a key contribution of our work.

III. HTTP-based Adaptive Streaming using MDP

In the literature, streaming rate adaptation has been modelled with MDP in various forms and complexities. In this section, we mainly follow the formulation of [5], [6] with some specific simplifications. We outline the key parameters of the optimisation framework central to the understanding of our proposed overhead reduction approaches:

System states and decision timings: We observe the system state when a video chunk is completely downloaded. The system state $S(\rho, q)$ is jointly represented by the quality level (q) of the downloaded chunk and the amount of time available (ρ) before its playback deadline. There is a deadline miss if the chunk download is not completed before its deadline ($\rho < 0$), in which case the video is frozen for a while until the chunk is downloaded, and it is played immediately at that time. Therefore, for a deadline miss, ρ is considered zero instead of negative.

If there is not enough space remaining in the buffer for another chunk after storing a downloaded chunk, the decision making and start of downloading the next chunk stall until there is enough room in the buffer. The value of ρ therefore assumes the value at the time of decision making (when there is space in the buffer) instead of when the last chunk was downloaded. This provides an upper bound for ρ, which is basically controlled by the buffer size. For example, if we have a buffer with a capacity to hold 7 chunks each 2 seconds long, then the upper bound for ρ is 14 seconds. Note that chunks have different sizes based on the quality while in most practical systems a buffer will have a maximum size in terms of bytes. One way to address this issue would be to configure the buffer size in bytes using the maximum chunk size, but measure buffer occupancy in units of chunks stored in the buffer.

Although ρ is a continuous number between zero and upper bound, we can use a discrete interval system to achieve a finite number of MDP states. We divide each second into n intervals and use an integer to represent the value of ρ. For example, for a 7-chunk buffer holding only 2-sec chunks, $n = 2$ would give us $7 \times 2 \times 2 = 28$ different values for ρ.

Actions: At each state, the decision taken is referred to as an action. For our adaptive HTTP streaming system, an action is basically a decision about the quality level for the next chunk. If we have N quality levels to choose from, then we have N possible actions. Each action will yield a different probability for completing the download of the next chunk at a specific time interval and hence specific value for ρ for the following state. Clearly, an action chosen (decision made) at the current state will influence the transition probability of reaching to a specific state at the next step.

Transition probabilities: Given the action taken, some transition probabilities will be clearly zero. For example, if the decision is to download the next chunk in quality level 3, then in the next step, we are only concerned with calculating the transition probabilities for states with quality 3; the transition probability to reach any state with quality level other than 3 would be zero. Given the size of a chunk is known, the probability that in the next step the system will arrive at a state with a specific value for ρ can be calculated using the cumulative distribution function (CDF) of the underlying network bandwidth as follows. For T-sec video chunks in N quality levels, and assuming an upper bound of M with n discrete intervals per second for measuring the value of ρ, the transition probability from state (i, x) to state (j, y) can be obtained using the following equation, which for $1 \leq q \leq N$, yields a 3D matrix of size

$$\{(M \times T \times n + 1) \times N\} \times \{(M \times T \times n + 1) \times N\} \times N:$$

$$P^q_{(i,x)(j,y)} =$$

$$\begin{cases} 0 & 1 \le x \le N, y \ne q, 0 \le i \le M \times T \times n, 0 \le j \le M \times T \times n \\ {P_i}^q_j & 1 \le x \le N, y = q, 0 \le i \le M \times T \times n, 0 \le j \le M \times T \times n \end{cases}$$

where P^q_{ij} is calculated as:

$${P_i}^q_j = \begin{cases} 0 & \text{if } \begin{cases} 0 \le i \le (M-1) \times T \times n \\ T \times n + i \le j \le M \times T \times n \end{cases} \\ P^q(T \times n + i - j) & \text{if } \begin{cases} 0 \le i \le (M-1) \times T \times n \\ 1 \le j \le T \times n + i \end{cases} \\ 1 - \sum_{x=1}^{T \times n + i - 1} P^q(x) & \text{if } \begin{cases} 0 \le i \le (M-1) \times T \times n \\ j = 0 \end{cases} \\ P^q_{((M-1) \times T \times n)(j)} & \text{if } \begin{cases} (M-1) \times T \times n < i \le M \times T \times n \\ 0 \le j \le M \times T \times n \end{cases} \end{cases}$$

and $P^q(x)$ is calculated as:

$$P^q(x) = \begin{cases} 1 - F(n \times S(q)) & \text{if } x\text{= 1} \\ F\left(\frac{n \times S(q)}{(x-1)}\right) - F\left(\frac{n \times S(q)}{(x)}\right) & \text{if } x > 1 \end{cases} \quad (1)$$

where $S(q)$ is the size of a chunk in quality q and $F()$ is the CDF of the underlying network bandwidth. Note that downloading of next chunk starts immediately if buffer is not full ($i \le (M-1) \times T \times n$), but delayed when the buffer is full ($i > (M-1) \times T \times n$). The amount of delay will vary depending on the current state (relative progress or the value of i), but the downloading of next chunk will commence as soon as the buffer has a space ($i = (M-1) \times T \times n$), i.e., it changes its status from full to non-full, irrespective of the amount of delay. This means that for all $i > (M-1) \times T \times n$, the transition probabilities are identical and they are equal to the ones with $i = (M-1) \times T \times n$. Consequently, $T \times n + 1$ rows of the transition probability matrix will be identical.

Revenue function: The *Revenue function* $R^q(i,x)$ uses some rewards and penalties to evaluate the outcome when action q is chosen at state (i,x):

$$R^q(i,x) = u(q) - d(i,q) - c(x,q)$$

where $u(q)$ is a reward to watch a chunk in quality q, $d(i,q)$ is a penalty if a deadline is missed, and $c(x,q)$ is a penalty for changing a quality level from the last chunk to the next. The deadline penalty can be derived as a function of the probability that the next chunk will miss its deadline:

$$d(i,q) = \left\{1 - \sum_{x=1}^{T \times n + i} P^q(x)\right\} \times D$$

where $\left\{1 - \sum_{x=1}^{T \times n + i} P^q(x)\right\}$ is the probability of missing the deadline (the probability that the next chunk does not arrive in any of the intervals before the deadline) and D a constant that we can use to tune the MDP model. We can reduce the number of deadline misses by selecting a large value for D, and vice versa. Finally, $c(x,q)$ is a penalty for a specific change of quality level and we can assign different penalties for difference types of quality switches. For example, the penalty for switching to a lower quality can be larger than that for switching to a higher quality to encourage higher average quality for a video session. Similarly, penalty for jumping multiple quality levels can be harsher than a smoother change in quality.

Once the rewards and penalties are assigned, we basically need to solve the optimisation problem that maximises the revenue function. Value iteration [18] is a well known algorithm for solving such optimisations and we used the MATLAB implementation of this algorithm [19] to solve all our MDP formulations in this paper. Once solved, the outcome of the optimisation is an optimal action for each given state. This set of actions is called the optimal policy or strategy, which is essentially a 2-column table. Given an MDP strategy, an HTTP-streaming client can simply make the decision about the quality of the next chunk by first observing its current state, i.e., the quality of the last downloaded chunk and the value of ρ, and then looking up a strategy table.

While MDP provides an effective framework to tune and balance different dimensions of adaptive HTTP streaming performance by adjusting the rewards and penalty parameters, it requires the client to solve the MDP problem first before the strategy table can be used to make the decisions. However, as we have seen in the preceding equations, the optimisation depends on the CDF of the underlying bandwidth. Depending on how the client obtains this CDF, the MDP may lead to high computation overhead for a mobile client. For example, if the client only uses its on-line observations to learn the CDF, then it may want to compute the MDP every time there is a new bandwidth observation to ensure that most accurate CDF is used in the optimisation. In the case of HTTP streaming, this means one MDP optimisation for every chunk download, as we obtain a new bandwidth observation when a new chunk is downloaded. However, this would lead to very high overhead. It is therefore important to consider implementation options that would reduce MDP overhead without significantly deteriorating its performance, which is the topic of the next section.

IV. MDP Strategy Update Approaches

In this section, we propose three MDP strategy update approaches, k-MDP, s-MDP, and x-MDP, that reduce MDP computation overhead in different ways. Specifically, these three approaches differ in ways they attempt to estimate the parameters of bandwidth CDF.

k-MDP: The first approach is called k-MDP, which is designed to reduce MDP overhead when the client uses only the online observations to estimate bandwidth CDF. The online bandwidth estimation is simple to realize in practical systems as it does not require any offline network measurement, which could be a costly and time intensive exercise. With online estimation, a bandwidth sample is obtained each time a video chunk is downloaded, which is basically derived from the chunk size and the time it took to download the chunk. Thus, with each chunk downloaded, the client has the opportunity to update the CDF parameters, such as the mean (μ) and standard deviation (σ) of the network bandwidth, to improve the accuracies of MDP transition probabilities. However, to

take advantage of the improved CDF estimation, the MDP optimization has to be recomputed. With k-MDP, the client updates the CDF parameters each time a chunk is downloaded, but recomputes MDP strategy only after downloading k chunks. The MDP computation overhead of this approach, therefore, can be controlled linearly by increasing/decreasing the value of k. A smaller k would provide a more accurate MDP strategy at the expense of higher computation overhead and vice versa. It is important to note that even with a small k, the online MDP would suffer from 'sample scarcity' problem in the beginning of the streaming session; the estimation would improve gradually as more chunks are downloaded.

s-MDP: One way to completely eliminate the online MDP optimisation overhead is to solve the MDP a *priori* using offline network measurement. s-MDP uses the global bandwidth statistics for a given region to generate an MDP policy which is used throughout a given trip taken in that region. With this approach, the client incurs no online MDP overhead and there is only a single offline MDP calculation. Needless to say that s-MDP is only applicable when bandwidth statistics for a given region is available.

x-MDP: Like s-MDP, x-MDP also incurs zero online overhead, but it computes separate MDP strategies for each x meters of a road using the bandwidth CDF specific to a given x-meter road segment. This approach is motivated by the previous findings that distribution of mobile network bandwidth may change significantly from one road segment to another even within the same region [10], [11]. With x-MDP, the client will switch to a new MDP strategy after travelling every x meters, but no online computation will be incurred as all of these strategies are precomputed offline and stored in a database. It is obvious that x-MDP is only applicable when road segment based bandwidth statistics are available. It is important to note that, unlike the online k-MDP, s-MDP and x-MDP do not suffer from sample scarcity as they rely on extensive offline network measurements. However, the offline MDP approaches implicitly rely on the fact that the CDF estimated from the offline statistics would accurately capture the bandwidth uncertainty exhibited during a real trip. In our simulation experiments (Section V), we considered offline bandwidth statistics measured from the same routes in previous trips to minimize differences between the learning and testing setups. In practice, this could be achieved if the client gathers its own offline statistics from previous video sessions along regularly travelled routes.

V. Simulations

In this section, we outline the simulation setup that we have used to evaluate the three approaches outlined in the previous section. We first provide an overview of the empirical bandwidth traces that we have used in the evaluations. Next, we briefly discuss the parameters of the video and the MDP model.

A. Empirical bandwidth traces

In our evaluations, we use real-world mobile bandwidth traces collected empirically by Yao et al. [10] . The researchers measured the downlink mobile bandwidth at approximately every 10s while driving along a route in the city of Sydney.

TABLE I: Illustrative Example of Bandwidth Traces

	time	latitude	longitude	bandwidth (Kbps)
1	1186549400	-33.919785	151.228913	1663.1440
2	1186549410	-33.919635	151.227787	1964.7330
3	1186549420	-33.91958	151.227322	2038.8659
4	1186549430	-33.91958	151.227322	2011.2631
5	1186549440	-33.91953	151.22692	1838.6578
6	1186549450	-33.91905	151.226322	1208.2767

TABLE II: Bandwidth statistics for 1000m road segments

Road segment	# of samples	μ	σ
1	825	442.07	249.05
2	1022	478.57	368.20
3	1206	423.60	92.66
...	...	...	...
23	758	438.87	126.37
24	79	444.14	94.08
Whole-Route	12716	438.02	251.61

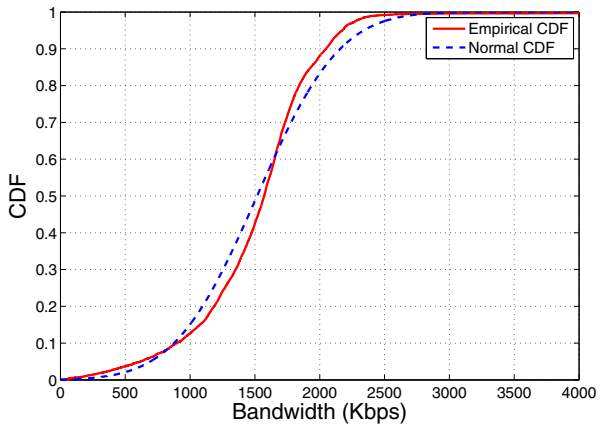
The route is 24 Km long and typical drive time ranges from 22 to 30 minutes. The bandwidth measurements were tagged with the GPS coordinates and time. Measurements were conducted simultaneously for two 3G providers. In this paper, we use the traces from one provider. Table I illustrates an example of 6 data points. Column one represents the time when samples are recorded, column two and three are the geographical coordinates and last column is the measured downlink bandwidth. 70 repeated trips were made along this route, resulting in a total of 70 bandwidth traces. We use the first 64 trips to generate the bandwidth statistics for the offline MDP approaches while the final 6 trips (65-70) are used for our evaluations.

Recall that the single MDP (s-MDP) scheme uses global bandwidth statistics from a given region. We assume that the entire trip is encompassed in a single region. Thus, we need to compute a single CDF for the whole route and then use this CDF to compute the transition probabilities and generate a single MDP strategy for the entire route. In Figure 1 (a), we plot the empirical CDF as well as the normal CDF. We find that normal CDF serves as a good approximation for this set of bandwidth samples. For all three approaches, we therefore compute only two parameters, mean (μ) and standard deviation (σ) to estimate the bandwidth CDF.

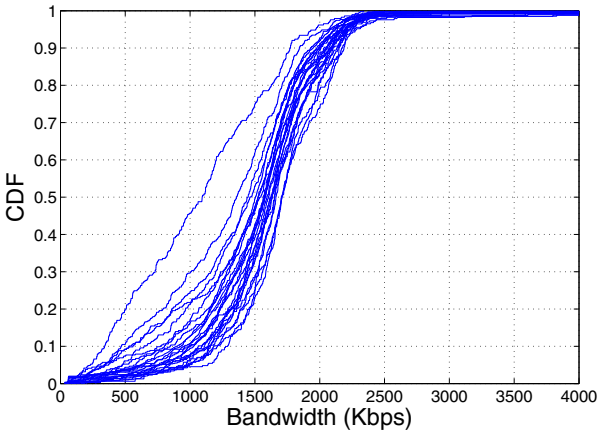
Unlike s-MDP, x-MDP recomputes the optimal strategy for every x meters of the road. We consider an $x = 1000$, which yields good number of samples for each road segments. Segment-wise statistics for the first 64 trips are shown in Table II. As we can see, with 1000-meter segments, we have 24 different μ's and σ's, which provide 24 different normal CDFs. Figure 1 (b) clearly shows that there are differences between CDFs of individual road segments, motivating the need for MDP recomputations at segment boundaries.

B. Video Statistics

In our evaluations, we used the Big Buck Bunny movie [20]. The original clip is only 9:56 min long, but we repeated the movie until the end of a trip. We created five different



(a)



(b)

Fig. 1: CDF of bandwidth traces (a) empirical vs. normal CDF, and (b) individual (normal) CDFs of 24 road segments

quality versions of the movie. We used ffmpeg [21] to encode the original video file using bit-rates of 186 kbps (*quality 1*), 499 kbps (*quality 2*), 1101 kbps (*quality 3*), 1292 kbps (*quality 4*) and 1898 kbps (*quality 5*). Figure 2 illustrates one particular frame of this video encoded at these 5 different quality levels. Each video stream was multiplexed with a common 128 kbps audio file to form a corresponding single MPEG-2 TS stream. The resulting stream is divided into 2 second chunks. We compute the average and variance of the chunk sizes for each quality of the movie. As observed in Table III, the variance is small, so we used the average chunk size in our simulations instead of exact sizes for each individual chunk (Equation 1 shows how chunk size is used in calculating transition probability).

C. MDP parameters

As explained in the previous sections, MDP is a flexible optimisation framework whose outcome can be influenced in any of the three dimensions of streaming performance by

Fig. 2: Quality levels

TABLE III: Mean, standard deviation, and coefficient of variance (Cv) of 2-sec chunk size (in Kb)

	$q1$	$q2$	$q3$	$q4$	$q5$
μ	375.29	938.77	2027.54	2360.88	3513.08
σ	10.91	122.22	255.82	351.84	874.84
Cv	0.03	0.13	0.13	0.15	0.25

adjusting its reward and penalty parameters. We keep the reward parameters fixed as shown in Table IV, but vary the deadline (D) and quality change (C) penalties. The value of D is varied between 2 and 350 and C is varied as a factor of the base values shown in Table V (we considered 10 different factors between 0.1 to 1.9). The values of other MDP parameters are as: N= 5, M= 7 and n= 2.

TABLE IV: Reward function

quality level (q)	**1**	**2**	**3**	**4**	**5**
$u(q)$	1	2	4	7	10

TABLE V: Base penalty values for changing quality level from i to j

$i \backslash j$	**1**	**2**	**3**	**4**	**5**
1	0	1	5	10	25
2	10	0	1	5	10
3	50	10	0	1	5
4	250	50	10	0	1
5	500	250	50	10	0

VI. Results

In this section, we analyse simulation results to evaluate and compare the performance of the three proposed MDP overhead reduction approaches, k-MDP, s-MDP, and x-MDP. Before the comparison, we first examine the effect of param-

TABLE VI: MDP streaming performance as a function of penalty values. The columns show different values of quality change penalty and the rows show the deadline miss penalties.

D\C	0.1	0.3	0.5	0.7	0.9	1.1	1.3	1.5	1.7	1.9
10	195.4	202.8	207.0	230.4	268.8	277.4	280.8	282.6	282.8	286.4
	4.74	4.74	4.75	4.77	4.80	4.81	4.81	4.82	4.81	4.81
	47.60	34.00	26.40	20.80	14.60	13.80	13.00	12.60	12.00	10.80
15	66.80	62.60	60.00	59.80	55.20	52.40	54.20	51.40	48.40	43.20
	4.58	4.57	4.57	4.56	4.55	4.54	4.54	4.52	4.50	4.44
	61.40	48.80	39.80	34.20	32.00	31.20	30.40	29.20	27.40	24.60
20	48.00	46.80	47.80	44.20	39.60	41.60	36.20	33.40	30.20	27.60
	4.55	4.55	4.54	4.54	4.53	4.52	4.47	4.43	4.37	4.32
	64.00	46.20	39.60	34.60	32.40	32.00	28.40	26.00	23.00	20.60
24	40.80	36.60	37.60	37.40	34.40	30.80	31.20	26.60	26.80	25.60
	4.53	4.53	4.53	4.52	4.50	4.45	4.41	4.34	4.30	4.25
	65.60	47.20	40.20	37.20	33.20	28.80	27.20	21.60	20.20	17.00
27	28.00	32.40	32.60	32.20	33.40	30.20	27.60	24.60	24.00	20.00
	4.51	4.52	4.51	4.50	4.47	4.41	4.33	4.29	4.27	4.21
	73.40	47.20	41.80	37.20	33.60	26.80	23.60	20.00	20.20	16.20
30	22.40	27.60	29.80	29.60	26.60	25.60	23.60	24.00	23.20	21.80
	4.50	4.50	4.50	4.47	4.42	4.35	4.30	4.28	4.23	4.20
	78.80	51.20	44.80	36.40	30.40	26.00	21.60	20.80	17.80	16.20
50	10.60	13.20	17.60	16.80	15.00	15.20	16.40	17.40	16.60	16.60
	4.44	4.42	4.35	4.27	4.22	4.20	4.18	4.17	4.16	4.15
	87.20	72.60	54.80	42.60	28.00	26.40	21.60	18.40	17.80	16.20
70	7.40	8.40	12.20	11.80	12.00	12.00	11.60	12.60	12.00	12.40
	4.41	4.36	4.26	4.19	4.18	4.15	4.14	4.14	4.12	4.11
	98.20	73.80	52.40	42.80	33.60	29.20	26.20	24.60	23.20	22.40
100	5.20	6.20	6.20	5.80	6.80	6.60	6.80	9.20	8.40	8.00
	4.37	4.26	4.20	4.14	4.12	4.11	4.10	4.09	4.06	4.05
	107.2	65.40	47.00	39.60	35.60	33.60	28.80	26.80	26.00	23.40
130	4.40	5.20	5.80	5.00	5.20	5.00	4.60	5.20	4.60	6.60
	4.31	4.22	4.14	4.12	4.10	4.09	4.08	4.04	4.02	4.03
	107.2	62.80	41.20	36.80	36.00	32.20	27.60	25.20	23.40	22.40
150	4.00	5.80	5.60	6.00	5.00	4.20	4.20	4.80	4.20	4.60
	4.28	4.18	4.12	4.11	4.09	4.08	4.05	4.01	4.01	4.02
	107.0	57.20	40.60	38.20	36.20	33.00	29.20	24.40	23.00	23.80
…	…	…	…	…	…	…	…	…	…	…

eter tuning in MDP for balancing the optimization over the three dimensions of video quality.

A. Parameter tuning in MDP

Table VI shows the streaming performance for the online MDP with k=1 for various combinations of deadline miss penalty D (rows) and quality change penalty C (columns). For each combination of D and C, the three real numbers represent the performance in three dimensions. The top number represents the number of deadline miss (DM) for the video session, the middle represents the average chunk quality (AQ), and the bottom represents the number of times quality was changed (QC) in the session. All these numbers are averaged over the five test trips.

The tuning opportunity with MDP is clearly demonstrated by Table VI. For example, by setting a high value for D, say D=150 (last row), one can keep the deadline miss to a minimum (only about 4 deadline miss for the entire trip). For the minimal deadline miss with D=150, one can either set a small value for C (say 0.1) to watch a high quality video (average quality level of 4.28), but with large number of quality changes (107), or the number of quality changes could be reduced significantly (to 23.8) for a slightly lower AQ of 4.02.

Figure 3 shows the flexibility of MDP using a scatter plot of all data obtained from many different combinations of D and C values. We can see that not only we have a wide range of options between the AQ (x-axis) and the DM (y-axis), we have the opportunity to trade-off between DM and QC when considering a specific AQ. See for example AQ values between 4 and 4.5. In this interval, we have different DM for the same AQ, but smaller DM is achieved with larger QC, as the system has to switch to a lower quality more often to avoid a potential deadline miss. We further observe that MDP yields a non-linear trade-off between picture quality and deadline miss with number of deadline miss increases rapidly if we try to watch

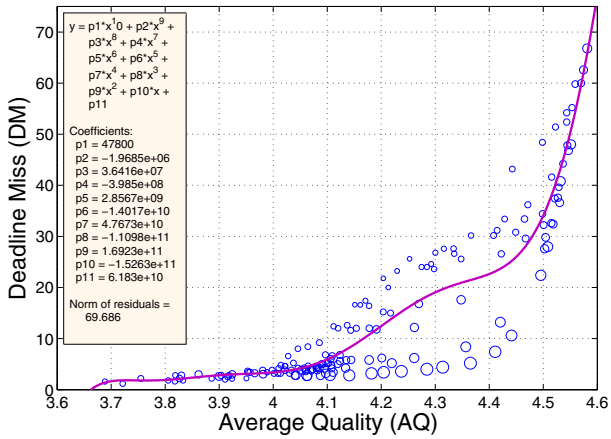


Fig. 3: Scatter plot of AQ and DM data. The size of the circles is adjusted to reflect the different values of QC with larger circles denoting larger QC, and vice versa. For some AQ values, we have a wide range of DM, where larger DM correspond to smaller QC, as seen in the middle of the graph between 4 and 4.5 in the x-axis.

TABLE VII: Length of simulation (in seconds) of k-MDP for different values of k. Simulation was run on a laptop with an i5-3320M-2.60GHz CPU and 8GB RAM

		k=1	k=10	k=20
Trip	65	502.79	53.74	26.80
	66	343.81	36.33	19.66

the video in very high quality. We were able to fit this non-linear trade-off to a tenth degree polynomial.

B. Online MDP

In this section, we evaluate the performance of k-MDP in terms of online computational overhead and streaming performance. Intuitively, with an increasing k, the proposed k-MDP would reduce online computation overhead linearly because the value of k directly controls the number of times the MDP optimization will have to be solved. To verify this, we have recorded the total simulation time for two different test trips for different values of k (see Table VII). We can see that the simulation time reduces as we increase k and the reduction is roughly proportional to k.

A key feature of k-MDP is the trade-off between computational overhead and streaming performance. To characterize this trade-off, we now turn to investigate the effect of an increased k on the streaming performance. Figure 4 plots the effect of k on all three dimensions by varying the deadline miss penalty. We find that the effect of k is most pronounced on deadline miss. In the subsequent analysis, we therefore focus

TABLE VIII: Streaming performance of k-MDP for different values of k

	k=1	k=5	k=10	k=20	k=37	k=50
DM	15.54	16.54	17.18	19.4	20.9	22.84
AQ	4.256	4.263	4.262	4.266	4.269	4.268
QC	38.56	37.76	36.28	36.62	36.66	36.64

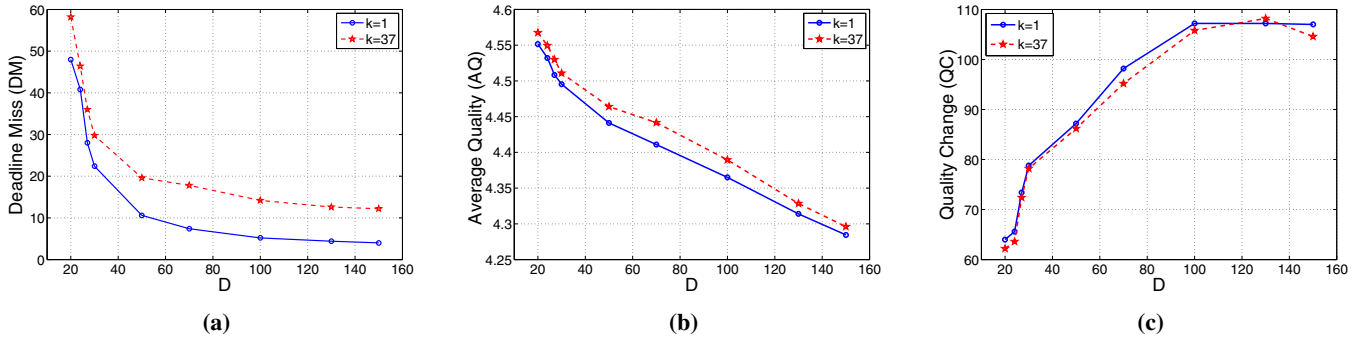


Fig. 4: Impact of deadline miss penalty on three dimensions of quality for two different values of k. $k = 37$ causes noticeable increase in DM compared to $k = 1$, but no noticeable difference in AQ and QC.

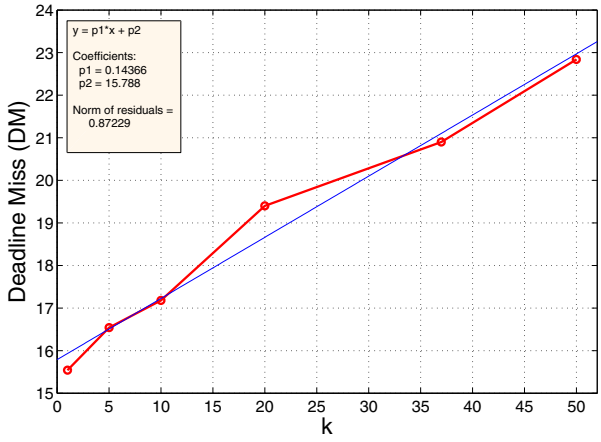


Fig. 5: Deadline miss increases linearly with k. There is roughly one extra deadline miss for every 10 jumps in k.

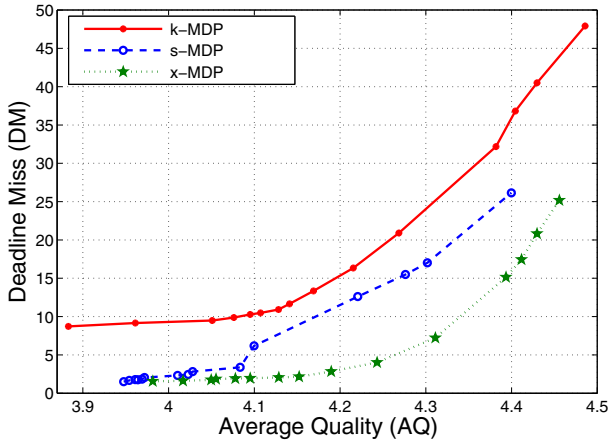


Fig. 6: Comparison between online MDP (k-MDP) and offline MDP (s-MDP and x-MDP).

on this dimension to find out how deadline miss is affected as a function of k.

Table VIII shows the streaming performance for different values of k for D=130 averaged over 10 different values of C in the interval [0.1, 1.9] with increments of 0.2. As we can see, k has a negligible effect on AQ and QC, but number of deadline miss continues to increase as we increase k. Figure 5 shows that deadline miss increases linearly with k. We have about one extra deadline miss for every 10 jumps in k. Note that each deadline miss represents a video freezing event during the session. Given that k has linear effect on both computation overhead as well as streaming performance (deadline miss), we can conclude that k-MDP yields a linear trade-off between performance and overhead.

C. *Offline MDP*

Note that off-line MDP approaches, s-MDP and x-MDP, have zero on-line overhead as all optimisations are computed off-line. We are interested in comparing their performances with those of k-MDP to see how the elimination of on-line overhead impacts performance. Since we found that MDP provides a non-linear trade-off between AQ and DM (Figure 3), it is important that we compare off-line approaches with on-line ones using this trade off curve.

For a given MDP approach, we derive the AQ-DM trade off curve by first generating a large number of performance data for many different combinations of deadline miss penalty and quality change penalty similar to the combinations shown in Table VI. We have run a total of 150 simulations representing 150 combinations, where C assumed 10 different values in the interval [0.1, 1.9] with increments of 0.2, and D assumed 15 different values in the interval [2,350]. For each value of D, we obtain a single performance value by averaging all performance data from 10 different values of C. This gives us 15 trade off data points for each approach as plotted in Figure 6. For k-MDP, we consider a value of k=37, as we find that on average about 37 chunks are downloaded in each of the 1000-meter road segment. This aligns k-MDP approach with x-MDP, which also switches to a new MDP strategy after every 1000 meters of travel.

We make several observations. First, we find that like on-line, off-line MDP also exhibits similar non-linear trade-off between AQ and DM. Second, we find that although on-line overhead is totally eliminated in the off-line approaches,

they actually perform better than on-line approach. x-MDP performs the best, which could be attributed to its ability to make use of more precise estimation of the CDF of mobile bandwidth.

VII. Conclusions

We have proposed and evaluated three approaches to reduce optimisation overhead for HTTP-based adaptive streaming when MDP is used as the underlying optimisation framework. For on-line optimisation, we have shown that updating the MDP strategy after downloading every k chunks yields a linear trade off between performance and overhead. Conceptually, on-line overhead could be completely eliminated with off-line optimization using only past observations of mobile bandwidth for a given region or road segment. Our simulation experiments involving real bandwidth and mobility traces along with actual video contents have revealed that such pure off-line MDP optimizations outperform the on-line optimization in terms of improved trade off for picture quality and deadline miss. The best performance is achieved when the MDP strategy is optimised using the bandwidth CDF of each specific road segment, which allows the system to realign with any statistical differences between different locations of a given region.

A possible reason for offline MDP outperforming the online-MDP could be because the statistics used for CDF estimation in offline MDP were more extensive (data from 64 previous trips) than the ones that could be used for the online MDP (data only from the current trip). Another reason for such improved performance with offline MDP optimization could be because all our data were collected along the same route. How much of these improvements could be retained if data from different routes, but in the same region, were used would be worth investigating as a future work.

Acknowledgement

We acknowledge the five anonymous reviewers whose detailed comments helped improving the quality of the final version of this paper.

References

[1] Apple, "HTTP Live Streaming Overview," [Online accessed 01-March-2013], URL: https://developer.apple.com/library/ios/#documentation/networkinginternet/conceptual/streamingmediaguide/Introduction/Introduction.html.

[2] A. Zambelli, "IIS smooth streaming technical overview," *Microsoft Corporation*, vol. 3, 2009.

[3] Adobe, "HTTP Dynamic Streaming on the Adobe Flash Platform," [Online accessed 04-March-2013], URL: http://www.adobe.com/au/products/hds-dynamic-streaming.html.

[4] T. Stockhammer, "Dynamic Adaptive Streaming Over HTTP: Standards and Design Principles," in *Proceedings of the second annual ACM conference on Multimedia systems (MMSys)*, San Jose, USA, 23-25 February 2011.

[5] C. C. Wüst and W. F. Verhaegh, "Quality Control for Scalable Media Processing Applications," *Journal of Scheduling*, vol. 7, no. 2, pp. 105–117, 2004.

[6] D. Jarnikov and T. Özçelebi, "Client Intelligence for Adaptive Streaming Solutions," *Signal Processing: Image Communication*, vol. 26, no. 7, pp. 378–389, 2011.

[7] S. Xiang, L. Cai, and J. Pan, "Adaptive Scalable Video Streaming in Wireless Networks," in *Proceedings of the 3rd Multimedia Systems Conference (MMSys)*, Chapel Hill, USA, 22-24 February 2012.

[8] M. Xing, S. Xiang, and L. Cai, "Rate Adaptation Strategy for Video Streaming over Multiple Wireless Access Networks," in *IEEE Global Communications Conference (GLOBECOM)*, Anaheim, 3-7 December 2012.

[9] J. Yao, S. S. Kanhere, and M. Hassan, "Improving QoS in High-Speed Mobility Using Bandwidth Maps," *IEEE Transactions on Mobile Computing*, vol. 11, no. 4, pp. 603–617, 2012.

[10] ——, "An Empirical Study of Bandwidth Predictability in Mobile Computing," in *Proceedings of the third ACM international workshop on Wireless network testbeds, experimental evaluation and characterization (MobiCom-WiNTECH)*, San Francisco, USA, 14-19 September 2008.

[11] P. Deshpande, X. Hou, and S. R. Das, "Performance Comparison of 3G and Metro-Scale WiFi for Vehicular Network Access," in *Proceedings of the 10th ACM conference on Internet measurement*, Melbourne, Australia, 13 November 2010.

[12] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video Streaming Using a Location-Based Bandwidth-Lookup Service for Bitrate Planning," *ACM Transactions on Multimedia Computing, Communications and Applications*, vol. 8, no. 3, p. 24, 2012.

[13] V. Singh, J. Ott, and I. D. Curcio, "Predictive Buffering for Streaming Video in 3G Networks," in *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, San Francisco, USA, 25-28 June 2012.

[14] I. D. Curcio, V. K. M. Vadakital, and M. M. Hannuksela, "Geo-Predictive Real-Time Media Delivery in Mobile Environment," in *Proceedings of the 3rd workshop on Mobile video delivery (MoViD)*, Firenze, Italy, 25 October 2010.

[15] P. de Cuetos and K. W. Ross, "Optimal Streaming of Layered Video: Joint Scheduling and Error Concealment," in *Proceedings of the eleventh ACM international conference on Multimedia (MM '03)*, Berkeley, USA, 02-08 November 2003.

[16] N. Mastronarde, K. Kanoun, D. Atienza, P. Frossard, and M. van der Schaar, "Markov Decision Process Based Energy-Efficient On-Line Scheduling for Slice-Parallel Video Decoders on Multicore Systems," *IEEE Transactions on Multimedia*, vol. 15, no. 2, pp. 268–278, 2013.

[17] V. Charvillat and R. Grigoraş, "Reinforcement Learning for Dynamic Multimedia Adaptation," *Journal of Network and Computer Applications*, vol. 30, no. 3, pp. 1034–1058, 2007.

[18] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley.com, 2009, vol. 414.

[19] Mathworks, "Markov Decision Process Toolbox," [Online accessed 20-November-2012], URL: http://www.mathworks.com.au/matlabcentral/fileexchange/25786-markov-decision-processes-mdp-toolbox.

[20] BlenderFoundation, "Big Buck Bunny," [Online accessed 04-March-2013], URL: http://www.bigbuckbunny.org.

[21] FFmpegProject, "FFmpeg," [Online accessed 04-March-2013], URL: http://ffmpeg.org.