CHAPTER 1

MECHANISMS FOR QUOTA-AWARE VIDEO ADAPTATION

JIASI CHEN[1], AMITABHA GHOSH[2], AND MUNG CHIANG[1]

[1]Princeton University, Princeton, New Jersey
[2]UtopiaCompression, Los Angeles, California[3]

1.1 Introduction

1.1.1 Two Conflicting Trends

Two recent and conflicting trends in Internet applications motivate the discussion in this work: video traffic becoming dominant, and usage-based pricing becoming prevalent.

Video traffic becoming dominant: Video consumption is on the rise. For example, Cisco predicts that 70% of all mobile traffic will be from video alone by 2016 [1]. Likewise, wireline traffic is also dominated by video, with NetFlix ac-

[3]Work was done when the author was a post-doctoral research associate at Princeton University.

counting for almost 30% of the wireline Internet traffic [2]. Together with YouTube, Netflix, Hulu, HBO Go, iPad personalized video magazine apps, and news webpages with embedded videos, video traffic is surging on both wireline and wireless Internet.

Usage-based pricing becoming prevalent: Tiered pricing, or usage-based pricing, is increasingly becoming commonplace in the U.S. as well as in other countries for both wireless and wireline broadband. Table 1.1 compares the usage fees from various international Internet Service Providers (ISPs).

Carrier	Country	Wireline/Wireless	Baseline Quota	Overage Charge
AT&T	USA	Wireless	2 GB	10 USD per GB
Verizon	USA	Wireless	2 GB	10 USD per GB
Reliance [3]	India	Wireless	2 GB	0.01 Rupee per 10 kB
Rogers [4]	Canada	Wireline	80 GB	2 CAD per GB
AT&T	USA	Wireline	250 GB	10 USD per 50 GB

Table 1.1 Usage fees for wireline and wireless ISPs around the world.

These two trends, *video traffic becoming dominant* and *usage-based pricing becoming prevalent*, are at odds with each other. On the one hand, videos, especially on high-resolution devices (e.g., iPhone 5, iPad 3, Android tablets), consume much more data than other types of traffic. For instance, 15 min of low bitrate YouTube video per day uses up 1 GB a month; likewise, one single standard-definition movie can take up to 2 GB. On the other hand, usage-based pricing threatens the business model of delivering entertainment via the high speed Long Term Evolution (LTE). These factors can result in high overage charges by the service provider, subscription to more expensive data plans, or even discontinuation of data service by disgruntled users. Given this conflict, a natural question to ask is: *Can a consumer stay within her monthly data quota without suffering a noticeable drop in video quality (distortion)?*

1.1.2 Current Approaches in Practice

In today's practice, there are two main approaches to balancing the competing goals of delivering high quality video while consuming less data.

Decrease video consumption through self-imposed warnings: Consumers may be warned by (a) service providers, or (b) self-imposed warnings to stop watching more videos once their usage-based charges become too high. In the first case, consumer concern over "bill shock" has resulted in legislation against service providers. For example, the European Union passed regulations in 2010 requiring mobile ISPs to provide warnings when a consumer has reached 80% of her quota, and to block data access once the usage fee reaches 50 Euro [5]. Similarly, the Federal Communi-

cations Commission (FCC) worked with commercial ISPs on a program to provide warnings to consumers when their data quota limits are reached [6].

Mobile applications that can report real-time data usage are also becoming increasingly prevalent. External applications, such as Onavo and DataWiz, provide consumers with attractive GUIs for monitoring data usage [7, 8]. Google also introduced a built-in data monitoring application in a recent version of the Android operating system [9]. Overall, however, these straight-forward warning systems can be undesirable, as they could result in dissatisfied users.

Decrease data consumption by lowering video quality: Content providers (CPs) can take a *"one size fits all"* approach of cutting back bitrates across all video requests, for all users, and at all times. For example, the YouTube mobile app automatically chooses a low quality video for requests made over a cellular data network, versus high quality video when a WiFi connection is used. Netflix implemented a similar approach in Canada in 2011 for both wireless and wireline customers in the light of expensive usage-based charges by Canadian ISPs. Netflix allows consumers to choose between three different qualities: low (0.3 GB per hour), medium (0.7 GB per hour), and high (1 GB per hour); however, this approach cuts down the quality across all videos, without taking user preferences and remaining quota into account.

1.2 Related Work

In this section, we give a brief background on existing video adaptation techniques and streaming protocols.

Video adaptation: Video adaptation is the process of modifying a video to suit the user preferences, usage environments, resource constraints, content characteristics, or digital rights, amongst other factors. In the context of this work, we consider video adaptation as a possible solution to the conflict between the growing demand for video traffic and usage-based data pricing. Adapting video quality with respect to resource constraints and user utility has been extensively studied since the 1990s, as surveyed, for example, by Chang and Vetra [10]. Liu *et al.* also considered video adaptation to maximize users' quality of experience (QoE), using the control knobs of video bitrate and Content Delivery Network (CDN) selection [13].

Video streaming protocols: In order to perform video adaptation, the appropriate technologies must be enabled in practice. The algorithms and systems to perform such video adaptation have long been studied in the research community [11, 12, 13, 14].

Traditionally, in practice videos were streamed using the User Datagram Protocol (UDP) due to their time-sensitive nature. More recently, video streaming over the Hyper Text Transfer Protocol (HTTP) has become common, due to HTTP's widespread acceptance, its ability to traverse Network Address Translations (NATs), and its support for caching at CDNs [15]. Various protocols, such as Scalable Video Coding (SVC), have also been proposed [16]. In SVC, each video is encoded into a base and multiple enhancement quality layers. Within the video, the video quality may be modulated.

However, SVC never gained widespread adoption, but increasingly interest from industry has focused on *dynamic adaptive HTTP video streaming over HTTP*. Apple, Microsoft, and Adobe have developed proprietary HTTP video streaming protocols that perform intra-video bitrate switching to adapt to varying channel conditions. These protocols are supported by various web technologies:

- Apple HTTP Live Streaming: Quicktime, iOS (iPhone, iPod Touch, iPad), Android
- Microsoft Smooth Streaming: Microsoft Silverlight, also adopted by Netflix
- Adobe OSMF: Adobe Flash

Recent progress by an industrial consortium has resulted in the MPEG-DASH standard [17], which aims to address the lack of interoperability between current video streaming protocols by providing an open interface to access the quality levels of a video. These lines of work still focus on modifying the video quality based on *channel* conditions, but underscore the increasing interest, both in academia and industry, in video adaptation technologies. Specifically, encoding and storing multiple bitrate versions of each video enables video quality adaptation with respect to each user's *data quota* in the context of usage-based pricing.

Quota-aware video adaptation: Jan *et al.* [18] developed a system for compressing images on webpages under a data quota. Although the motivation of addressing quota concerns is similar, its application is only to images and webpages. Recently, systems like Onavo [7] enables users to save on data plans by forwarding all images and text data through a proxy server where they are compressed. However, they do not exploit consumer usage patterns or deal with video traffic, which has a much larger range of compressibility (than images and text) and comprises the bulk of mobile data consumption. These could bring in additional substantial data saving opportunity.

1.3 A Potential Solution: QAVA

1.3.1 Trading off Quality vs. Cost vs. Volume

As a potential solution to the problem of video consumption under capped data plans, we propose the QAVA (Quota-Aware Video Adaptation) system. Our premise is the following: *Not every video bit is needed for every consumer*, and the bitrates can be adjusted not only based on screen resolution and channel conditions, but also usage patterns. QAVA can be customized to each user's demand and monthly data quota by adaptively choosing an appropriate bitrate for each video request, thereby *shaping the supply* for the user. We will show that by leveraging *video compressibility* and profiling *usage behavior*, QAVA can significantly mitigate the conflict between the growing demand for video traffic and usage-based pricing.

At the heart of QAVA is a *Stream Selector* (SS), which takes inputs from a *User Profiler* (UP) and a *Video Profiler* (VP), to select a particular bitrate and pre-emptively

compress the more compressible videos early in the billing cycle. The VP provides information related to a video, such as its compressibility, which measures the extent to which the size of a video can be reduced without a significant distortion in quality. The UP predicts consumer usage patterns from past history, and customizes the system to every user's flavor for watching certain types of videos. The SS then uses the information provided by both VP and UP to optimize QAVA for each user based on her monthly data quota.

The benefits to a QAVA-enabled user include the ability to watch more videos under a given monthly data plan without suffering a noticeable distortion in video quality, as compared to a non-QAVA user. Or, phrased differently, if a user's demand for video traffic remains the same or goes down, QAVA tries to save money for the user with a minimum impact on video quality. This 3-way tradeoff is illustrated in Figure 1.1. Across the three competing goals of minimizing cost, maximizing the number of videos watched, and minimizing distortion, QAVA strikes a graceful, tunable tradeoff.

1.3.2 Incentives for Players in QAVA Ecosystem

A natural question to our proposed approach is: *What are the incentives for different players in the ecosystem to use QAVA?* We address this from the perspective of three major players: users, ISPs, and content providers.

Users: A user has the most obvious incentive, because QAVA enables her to stop worrying about her monthly data plan and watch all the videos she wants with minimal distortion.

ISPs: An ISP has two options: it may wish to (a) reduce data traffic to lower network congestion and, thereby, cut down on operational and capital expenditure costs, or (b) preserve traffic to continue receiving overage charges from customers and/or usage fees from content providers. In the first case, QAVA ensures that all customers remain below their quota, which indirectly lowers the traffic rate. In the second case, the ISP can set quota parameters to ensure that its users still consume the same amount of data as non-QAVA users, but receive better video quality.

Content providers: The advantages for a content provider are three-fold. Firstly, since QAVA allows a user to access more content under the same data plan, the content provider achieves a greater profit from advertising revenue by increased content consumption. Secondly, the content provider improves customer satisfaction by removing her worries about exceeding the quota, which can be marketed as a competitive advantage over other content providers. Thirdly, QAVA reduces the potential need for the content provider to pay the ISP for customer data charges.

QAVA is thus mutually advantageous from the perspectives of all three players. In the remainder of this work, we will discuss QAVA's contribution along four different dimensions:

- **Architecture:** We design a modular system architecture for QAVA comprising three functional modules: SS, UP, and VP. Our design makes QAVA a deployable system that can be used by real-world consumers.

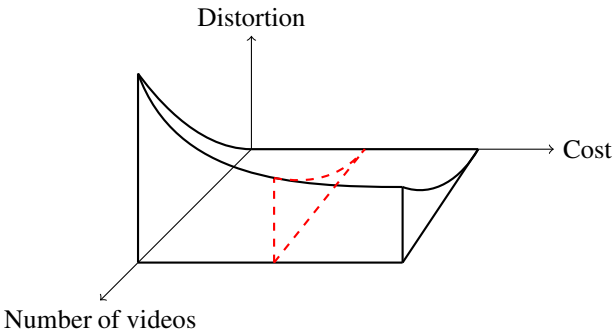


Figure 1.1 A 3-way tradeoff between distortion, cost, and the number of videos watched. For a fixed cost, QAVA enables a user to watch the desired videos while minimizing distortion.

- **Algorithms:** We design an online bitrate selection algorithm for the SS module based on the concepts of finite-horizon Markov Decision Process (MDP) [21]. This algorithm runs at the heart of QAVA, enabling it to provide a graceful, tunable control of the quality, cost, and volume tradeoff.
- **Experiments:** We evaluate the performance of QAVA in simulations using traces of real video requests.

1.3.3 Design Considerations

There are several alternatives and variants for designing the QAVA architecture. We briefly describe these alternatives, their advantages and disadvantages, and our design decisions.

Availability of Video Versions: We assume that all the videos are pre-encoded and stored on the content provider's server. An alternative to this is on-the-fly transcoding and adaptation, which requires compressing each video dynamically at the particular bitrate determined by the SS module. This can be a time-consuming operation and has significant implementation challenges; however, it would be capable of adapting video feeds for live events. In contrast, pre-encoded streams can be selected with minimal computation, but cannot handle video streams of live events.

We also assume that the different versions of a video chosen by the SS module are supported by the channel in terms of bandwidth requirement. These sustainable video versions may be pre-selected by the content provider based on typical wireless or wireline bandwidth, or chosen on-the-fly based on bandwidth estimation techniques currently proposed for use in adaptive HTTP video streaming algorithms [14].

Time Scale of Video Adaptation: There are two choices for the time scale of video adapation: (a) inter-video adaptation, and (b) intra-video adaptation. Inter-video adaptation is choosing a single bitrate stream for the entire duration of the requested video, whereas intra-video adaptation involves dividing each video into smaller clips and choosing the correct adaptation operation for each clip.

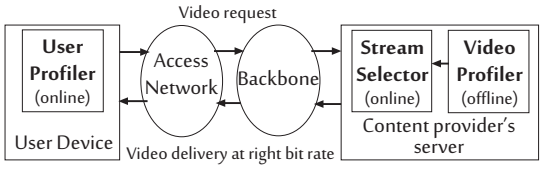


Figure 1.2 QAVA's modular system architecture: The UP module sits on a user device, whereas the SS and VP modules are located on a content provider's server. A video request originating from a user device travels through the access network and the backbone to the server, which then runs a stream selection algorithm to choose an appropriate bitrate to deliver to the user.

Inter-video adaptation is suitable for video clips of short duration (e.g., Youtube videos of less than 5 minutes), because the spatial and temporal activities tend to be similar throughout the duration. However, for longer videos such as movies, it is more appropriate to stream different bitrate versions for different parts of the video depending on the spatial and temporal activities. The algorithms presented in this work apply equally to inter- or intra-video rate switching. QAVA can be used for intra-video adaptation by considering each smaller segment as a separate video request. Such intra-video switching requires synchronous bit stream switching, which can be achieved with the advent of new video streaming protocols such as MPEG-DASH [17]. QAVA can also work with existing channel-based switching algorithms by optimizing and restricting the rates available to those algorithms.

Heterogeneous Data Quota: Data usage under a single data plan can be decomposed into three usage layers: (a) multiple users, (b) muliple devices per user, and (c) multiple traffic types per user per device. QAVA's control knob is on video traffic per user per device; thus, the "video quota" per device must be set. To accomodate non-video data traffic, the video quota should be set to a percentage of the total data quota based on historical video data usage. Running QAVA per user is also possible by aggregating video request logs across devices. This results in coarser granularity user profiling, but may improve the performance by decreasing sensitivity to noise. For the remainder of this work, we focus on the case of a single fixed video quota and a single user with a single device, but QAVA can easily be extended to encompass the other cases just outlined.

1.4 QAVA System Design

1.4.1 A Modular Architecture Design

The architecture of QAVA comprises three different modules, each one responsible for a specific function. The modules work together to enable QAVA optimize across the three performance goals shown in Figure 1.1. We first describe the motivation for each of the modules.

Module	Input	Output	Frequency
Stream Selector (SS)	Compressibility of video request, remaining monthly budget, and the user profile.	Selected bitrate version to deliver for the video request.	Every request
User Profiler (UP)	Time stamps and compressibility all past video requests.	Probability of a video request arriving at a time interval, and the video compressibility distribution for all past requests. Together these comprise the user profile.	Every billing cycle
Video Profiler (VP)	All videos stored in the content provider's server.	Compressibility of all videos.	Offline

Table 1.2 Three key functional modules of QAVA with their inputs, outputs, and update frequency.

Selecting Right Bitrates: The basic operation of QAVA is to choose an appropriate bitrate for every video request made by a user. This bitrate selection is based on two factors: (a) the user's data consumption pattern, and (b) the particular video requested. This job is performed by a *Stream Selector* module running at the heart of QAVA on the content provider's server, as shown in Figure 1.2. Due to space limitation, we focus on the pre-encoded bit stream scenario, where each video has multiple copies, each copy pre-encoded in a different bitrate and stored on the content provider's server. The number of copies with different bitrates of a video is pre-determined by the content provider.

Profiling Usage Behavior: A user's past data consumption history gives an indication of her future usage pattern. Since the video requests from a user can arrive at different times without any prior knowledge, a prediction of these arrival times is helpful to QAVA so it can choose the appropriate bitrates to serve the requests. A simple usage pattern could be watching on average x_d number of videos (or, equivalently, y_d bytes) on day d of a week. When the number x_d (or y_d) remains approximately the same for the same day d across different weeks, we may notice a weekly pattern. More complex usage behavior can lead to small-scale seasonal variations (daily or hourly) as well as *trends*, which are long term variations due to habitual changes that lead to a steady increase or decrease in data usage over months or years.

QAVA employs a *User Profiler* module to find patterns in usage behavior and to predict future video requests. In particular, the UP module estimates the probability of a video request arriving in a given time interval. The length of this interval can be uniform or variable depending on the past data consumption history, and is config-

ured by a system parameter. We design the UP module as an application that can be installed on a user device (client), as shown in Figure 1.2.

Estimating Video Compressibility: In addition to staying within a monthly data quota, the SS algorithm also aims to minimize video distortion. For this, the algorithm needs to know by what extent the requested video can be compressed and how much distortion it would cause in doing so. Different videos have different levels of compressibility depending on their spatial and temporal activities as well as on the choice of encoders. For example, a talk show that has very little motion in it can be greatly compressed using an H.264/AVC encoder, whereas a motion-rich sports video may not be compressible to the same extent.

The SS algorithm should be careful in choosing the right bitrate for every video request to avoid the following undesirable situation. Suppose the algorithm chooses a high bitrate for an easily compressible video when the user has a lot of quota left, possibly in the beginning of a month. Then it might be forced to choose low bitrates for some not-so-easily compressible videos near the end of the billing cycle in order to stay within the monthly budget, thus causing significant distortion. A possible remedy is to choose low bitrates for easily compressible videos *even when* there is sufficient quota left. However, such intelligent online decisions can be made only if the system knows about the distortion versus bitrate tradeoff for every video and can learn the quota consumption pattern over a billing cycle for each user. QAVA employs an offline *Video Profiler* module to compute this distortion for every bitrate and store it on the content provider's server, as shown in Figure 1.2.

We now summarize the three modules of QAVA:

- **Stream Selector (SS):** The SS module is at the heart of QAVA and is located on the content provider's server. It is an online module which decides on the right bitrate for every video request.

- **User Profiler (UP):** The UP module predicts the probability of future video requests from past usage history, and also computes a user-specific distribution of video types reflecting a user's taste of watching different types of videos. It is an online module and is located on the user device.

- **Video Profiler (VP):** The VP module is also located on the content provider's server and is an offline entity. It computes the distortion for every bitrate version of all the stored videos. We loosely call this the "*compressibility*" of the videos.

Other types of module separation and interconnections are also possible. For example, the VP and SS modules can be combined into a single module performing joint optimization. The stream selector requests videos of a certain compressibility, and the video profiler optimizes over various codecs to generate videos with desired characteristics. This would provide a finer decision granularity to the stream selector, but is computationally complex since the video profiler also runs video encoding operations.

1.4.2 Module Placement

In this section, we discuss the location of the various modules in QAVA. Although we refer to the specific modules of QAVA, our discussion of SS module placement is applicable to any general video adaptation system that makes online decisions.

The placement of the UP and VP modules is fairly intuitive: by necessity, the VP module is located on the content provider's server, since only the content provider knows about the video characteristics. Profiling the video on the user device is not feasible due to CPU and battery limitations. The UP module logs user data, so it should be placed on the user device to alleviate user concerns over data collection and privacy. The placement of VP and UP modules is shown in Figure 1.2.

1.4.2.1 Content Provider- or ISP-based Architectures The location of the SS module that runs the bitrate selection algorithm is, however, not so intuitive. For every video request, the SS module requires inputs from both UP and VP. One possibility is to place the SS module in the ISP's access network. Then, in order to satisfy a video request, the SS module first needs the video compressibility information to be sent from the content provider, as well as user information from the UP module. After receiving these inputs, the SS module runs the SS algorithm to choose the right bitrate. It then sends another request to the server to transmit the actual video in the selected bitrate. Overall, this results in unnecessary messages and potential delay, which is undesirable for delay-sensitive traffic such as video.

Alternatively, the SS module could be placed on the content provider's server. In order to satisfy a video request, the SS module still needs user information from the UP module, but already has access to information from the co-located VP. The selected bitrate can thus be transmitted immediately after the SS algorithm is run. Since this reduces the amount of message passing, placing the SS module on the content provider is more desirable. This also makes QAVA complementary to other video adaptation approaches [13].

Placing the SS and VP modules on the content provider's server incurs some monetary cost to the content provider, which must be overcome by the advantages discussed in Section 1.3.2. We argue that the cost to the content provider is small: it must install the SS module on its server (one-time), and compute the video profiles of all videos (a small amount of text data compared to video data size). And regardless of where the SS module is placed, our algorithms are equally applicable.

1.4.2.2 Client-based Architectures In case QAVA operates without the support from a content provider or an ISP, *client-based* architectures are possible. We discuss two potential architectures: (a) a transcoding-based system, and (b) a throttling-based system.

In a transcoding-based solution, all traffic from the client is routed through a web proxy. By inspecting the packets, video traffic can be distinguished and transcoded on-the-fly by the proxy server to the correct bitrate. This approach has the advantage of using standard web proxy technology, and can handle all types of videos as long as the transcoder has the appropriate codecs. However, there are significant implementation challenges in terms of latency minimization: the proxy and transcoder

must perform quickly in order to satisfy the demands of delay-sensitive video traffic. Moreover, implementing a transcoder for HTTP video streaming is also difficult.

A throttling-based solution leverages the emerging popularity of adaptive HTTP video streaming [17]. In this technology, the videos automatically adjust their bitrates based on their estimates of TCP throughput. By limiting the bandwidth observed by the rate switching algorithm, QAVA can force the video bitrates to automatically settle to the rates determined by the SS module. This throttling can be performed on the client device or on a web proxy. In terms of implementation, this approach is simpler than the transcoding-based solution, but suffers from the limitation that only adaptive HTTP video streams can be modified in this way.

1.4.3 QAVA Operational Example

In this work, we focus on the case where the SS module is located on the content provider's server. The following description does not change significantly for other placement options. The relationship between the different modules describing their inputs, outputs, and update frequencies is shown in Table 1.2. The input to the SS module is the compressibility of the requested video, the user's remaining monthly budget, and the user's profile as determined by the UP module. For every request, it runs the stream selection algorithm and outputs the selected bitrate version. The input to the VP module is the set of all videos stored on the content provider's server, and its output is the compressibility of each video. The input to the UP module is the time stamps for the user's past video requests, as well as the compressibility of those videos. Its output is the predicted video request probability and the video type distribution (i.e., compressibility distribution) specific to that user. These quantities characterize the user's behavior and are fed to the SS module. These predictions can be made in the beginning of the billing cycle, or updated more periodically throughout the billing cycle.

To be concrete, we give an operational example.

1. The content provider computes and stores the compressibility of all the videos on the server.

2. In the beginning of a billing cycle, the UP makes predictions (to be used in the current cycle) of the video request probability and the compressibility distribution based on its log of past requests of a user.

3. When a user requests a video in her current billing cycle, the request is sent to the SS module, which selects the bitrate to be delivered. The content provider also sends the compressibility of the requested video to the UP. The UP logs this as well as the timestamp of the request.

4. Once the current billing cycle is over, the UP updates its predictions based on the recent request logs. Steps 2–4 repeat for the next billing cycle.

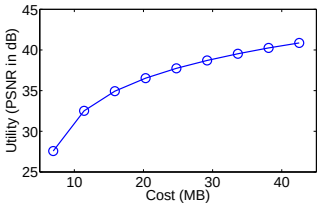


Figure 1.3 Video utility versus cost showing diminishing returns for increasing cost. The x-axis represents the size of the video when encoded from 100–900 Kbps in 100 Kbps increments.

1.5 Stream Selection

In this section, we first describe the video request, utility, and cost model, and then formulate the bitrate selection problem. We also introduce the key notation, as summarized in Table 1.3.

1.5.1 Video Request, Utility, and Cost Model

We divide the length of a billing cycle (e.g., month) into T time intervals, indexed by $t = 1, \ldots, T$, and assume that a user has a total budget B (measured in bytes) in one billing cycle. In each time interval t, a video request arrives with a certain probability, denoted by p_t. The remaining budget of the user at time t is b_t. The request probability p_t and budget b_t are provided by the UP module for each user.

Each video is encoded into M different bitrates, indexed by $j = 1, \ldots, M$. Associated with each video request arriving at time t, is a vector of utilites $\mathbf{u}_t = (u_{t1}, \ldots, u_{tM})$, and a vector of costs $\mathbf{c}_t = (c_{t1}, \ldots, c_{tM})$ for different bitrate versions of the video.[1] When there is no video request arrival at time t, the vectors $\mathbf{u}_t$ and $\mathbf{c}_t$ are null vectors with all components being zero, because then no bitrate is selected. The VP module provides the utility and cost of each video, $\mathbf{u}_t$ and $\mathbf{c}_t$, respectively.

Each user might prefer different types of videos with different compressibilities. For example, she might want to watch news clips that have different compressibilities than sports videos. To capture this effect, we introduce a joint probability distribution $P(\mathbf{u}, \mathbf{c})$, which is user-specific and represents the probability that a user requests videos with certain utility–cost characteristics. This distribution is provided by the UP module for each user.

In Figure 1.3, we show a typical utility versus cost function for a video encoded using the H.264/AVC codec with a resolution of 720×480 pixels. Such utility–cost curves are usually concave with diminishing returns for utility at higher cost (or equivalently, higher bitrate, since bitrate is proportional to data size for a fixed-length

[1]We note that these utility and cost vectors are fixed and not time-dependent. The use of the time index t in u_{tj} and c_{tj} is purely for the ease of exposition.

Symbol	Meaning
u_{tj}	Utility of bitrate version j for a video request arriving at t.
c_{tj}	Cost of bitrate version j for a video request arriving at t.
x_{tj}	Indicator variable (1 if bitrate j is chosen for video request at t; 0 otherwise).
M	Number of bitrates for each video.
T	Number of time intervals in a billing cycle.
$P(\mathbf{u}, \mathbf{c})$	User-specific joint probability distribution of video types based on past history.
p_t	Probability of a video request arrival at t.
b_t	Remaining user budget at t.
B	Total user budget in one billing cycle.

Table 1.3 Table of key notation.

video). A video with a flat utility-cost curve is "easily compressible," because lowering the bitrate decreases the utility only slightly. In contrast, a "hard-to-compress" video has a steep curve. We measure the utility u_{tj} of bitrate version j as its *peak signal-to-noise ratio* (PSNR) $\times$ the duration of the video [19]. The cost c_{ij} is the size in bytes. Discussion of the utility and cost metrics is reserved for Section 1.6.2.

1.5.2 Stream Selection as Knapsack Problems

We now formulate the problem of choosing the right bitrate by the SS module as different versions of the well-known knapsack problem [20] studied in combinatorial optimization. We first present an offline formulation, which is easy to understand, and then motivate the need for an online stochastic formulation.

1.5.2.1 Offline Multiple-Choice Knapsack Problem The goal of the stream selector is to choose the right bitrate for every video request made by a user in a single billing cycle. We aim to maximize the sum of the utilities across all video requests without exceeding the user's quota. In other words, we maximize the *average* video utility. An alternative formulation is to maximize the minimum utility across all videos requested during the billing cycle. However, since the high-level goal of QAVA is to maximize the overall user satisfaction, we optimize the average utility over time instead of the worst-case experience, as in the alternative formulation.

For a video request arriving at time t, we define a decision vector $\mathbf{x}_t = (x_{t1}, \ldots, x_{tM})$, where each x_{tj} takes the value 1 if bitrate version j is chosen, and 0 otherwise. Then

our problem is:

$$\begin{aligned} \text{maximize} \quad & \sum_{t=1}^{T} \sum_{j=1}^{M} u_{tj} x_{tj} \\ \text{subject to} \quad & \sum_{t=1}^{T} \sum_{j=1}^{M} c_{tj} x_{tj} \leq B \\ & \sum_{j=1}^{M} x_{tj} = 1, \ \forall \ t \\ \text{variables} \quad & x_{tj} \in \{0, 1\}, \ \forall \ t, j, \end{aligned} \tag{1.1}$$

where the first constraint says that the cost of the selected bitrates for all the videos requested in a billing cycle must not exceed the total budget B, and the second constraint says that only one bitrate version may be selected for each video.

This optimization problem is known as the Multiple-Choice Knapsack Problem (MCKP) [20]. In the regular single-choice knapsack problem, we are given a set of items, each with an associated value and weight. The objective is to pick a subset of the items such that the total value is maximized while not exceeding the knapsack capacity. In our stream selection problem, the items are the individual bitrate versions of the videos, and the multiple choices arise because exactly one version of each video must be selected.

The traditional offline version of MCKP, where all the input items are known in advance, is well-studied. The problem is NP-hard, but pseudo-polynomial time dynamic programming (DP) solutions exist [20]. Contrary to this offline version, the SS module does not know the video requests in advance, and so needs to make decisions in an online fashion. This requires a modification to the above formulation to handle online requests.

1.5.2.2 Online Stochastic Knapsack Problem

Unlike the traditional offline MCKP, in our scenario, the video requests are revealed one-by-one in an *online* fashion. Thus, existing DP solutions to the offline knapsack problem cannot be used. Online algorithms handle this situation by making a decision on-the-fly when a new video request arrives, without any prior knowledge of future requests. However, once a decision is made, it cannot be revoked or changed in the future.

Since the data quota is reset after a billing cycle is over, there is a time deadline before which all the actions must be made. We also note that the bitrate decisions for future intervals should not depend on the decisions taken at previous intervals, given the current remaining budget. This implies the Markov property. The problem can naturally be modeled as a finite-horizon Markov decision process (MDP). A key assumption is that the video requests are independent of time and, therefore, the transition probabilities are stationary.

The MDP formulation allows the SS module to make foresighted bitrate selection decisions by taking into account the future impact of its current decisions on the long-term utility. This is better than just an online algorithm which makes myopic

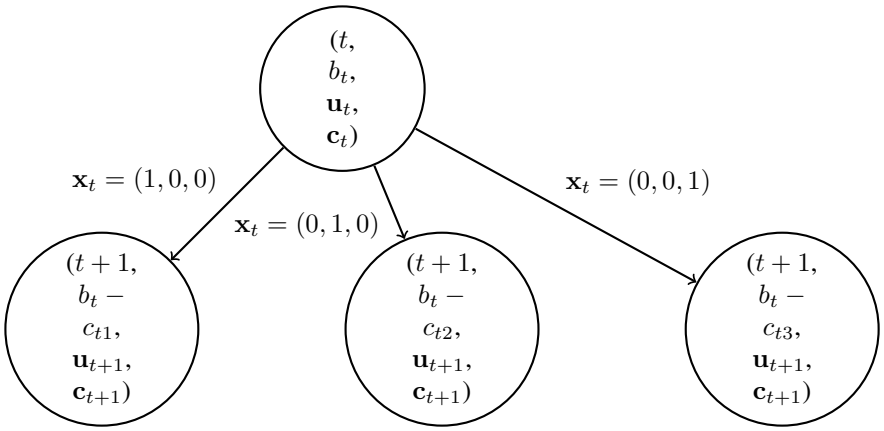


Figure 1.4 Stream selection modeled as a finite-horizon Markov decision process. A one step state transition is shown with 3 bitrate choices.

decisions at every time step. For example, a greedy solution might choose a bitrate that maximizes only the utility of the current request without overshooting the quota.

Figure 1.4 shows a simple example of choosing between three different bitrates over one time step. The state of the system is defined as the four-tuple $s_t = (t, b_t, \mathbf{u}_t, \mathbf{c}_t)$, comprising the current time interval t, the remaining quota b_t, and the utility and cost vectors $\mathbf{u}_t$ and $\mathbf{c}_t$. There are three possible actions: (a) choose the lowest bitrate, i.e., $\mathbf{x}_t = (1, 0, 0)$; (b) choose the second bitrate, i.e., $\mathbf{x}_t = (0, 1, 0)$; or (c) choose the third bitrate, i.e., $\mathbf{x}_t = (0, 0, 1)$. If the lowest bitrate is chosen, the system moves to time $t + 1$ with remaining budget $b_t - c_{t1}$. The algorithm collects utility (reward) u_{t1} and receives the new video request with utility and cost vectors $\mathbf{u}_{t+1}$ and $\mathbf{c}_{t+1}$. If the second bitrate is chosen, the system moves to time $t + 1$, but now subtracts the cost c_{t2} from its remaining budget, leaving it with $b_t - c_{t2}$. It also collects utility u_{t2}. A similar state transition results from choosing the third bitrate.

The set of actions $\{\mathbf{x}_1, \ldots, \mathbf{x}_T\}$ taken by the algorithm at every time step is called a *policy*. A policy that solves the MCKP of (1.1) is called an optimal policy. If the arriving video requests are known, an optimal policy can be determined using the traditional offline techniques previously mentioned. However, since the video requests are not known *a priori*, the MDP finds a policy that instead maximizes the *expected* sum of the utilities. We develop a solution using DP and online optimization.

1.5.3 Solving Finite-Horizon Markov Decision Process

The optimal policy can be found using the well-known backward induction techniques for finite-horizon MDPs [21]. We first define $U_t(b_t)$ as the expected utility accumulated from time t until the end of the billing cycle at time T, when the remaining quota is b_t. This expected utility assumes that the optimal action is applied

in each state. Then, the optimal action at each time step t is found by solving:

$$\begin{aligned} \text{maximize} \quad & u_{tj}x_{tj} + U_{t+1}(b_t - c_{tj}x_{tj}) \\ \text{subject to} \quad & c_{tj}x_{tj} \leq b_t \\ & \sum_{j=1}^{M} x_{tj} = 1 \\ \text{variables} \quad & x_{tj} \in \{0,1\},\ \forall j, \end{aligned} \tag{1.2}$$

where the first constraint ensures that the cost of the selected bitrate is less than the remaining quota. The objective function has an intuitive meaning: It maximizes the current utility plus the sum of the expected utilities, subject to the remaining quota. The problem can be solved in $O(M)$ time by discarding the bitrates that violate the constraints, and then picking the bitrate j^* that maximizes the objective function. It is solved every time a video is requested.

Solving (1.2) requires the computation of $U_t(b_t)$. It can be shown that the solution can be found by dynamic programming using:

$$U_t(b_t) = p_t\left(u_{tj^*} + E_{(\mathbf{u},\mathbf{c})}\left[U_{t+1}\left(b_t - c_{tj^*}\right)\right]\right) + \left(1 - p_t\right)U_{t+1}(b_t). \tag{1.3}$$

with boundary condition is that the expected accumulated utility is 0 when the billing cycle is over, or the remaining budget is less than 0 [30]. In this work, we choose the budget granularity to be 1, so b_t takes on possible values $1, \ldots, B$. The running time of computing the $U(\cdot)$ matrix is $O(TBM\Gamma)$, where Γ is the cardinality of the set $\{(\mathbf{u}, \mathbf{c})\}$ in the video type distribution.

The online and offline components of the MDP stream selection algorithm are summarized in Algorithm 1.1. In the special case of two bitrates ($M = 2$), our algorithm reduces to that of Papastavrou *et al.* [22]. With accurate user and video profiling, Algorithm 1.1 maximizes the sum utility while staying under the quota. In the case of inaccurate inputs, however, the algorithm may exceed the quota. In that case, the algorithm should simply choose the lowest bitrate, although this case never occurs in our numerical simulations.

Algorithm 1.1

MDP Stream Selection Algorithm

DP Computation of Utility Matrix

Input: Video type distribution $P(\mathbf{u}, \mathbf{c})$, quota B, and billing cycle length T.

Output: A matrix $\mathbf{U}$ of size $T \times B$.

1. Initialize $U_t(b_t) = 0, \forall\, t > T, b_t < 0$.
2. Compute each entry $U_t(b_t)$ of $\mathbf{U}$, using (1.3).

Online Bitrate Selection

Input: Utility--cost vectors $\mathbf{u}_t$ and $\mathbf{c}_t$, remaining budget b_t, quota B, billing cycle length T, and matrix $\mathbf{U}$.

Output: Optimal bitrate j^*.

if $b_t < 0$
  Choose $j^* = 1$.
else
  Discard the bitrates with cost greater than b_t.
  Compute j^* that maximizes the objective function of (1.2).
end

Update $b_{t+1} = b_t - c_{tj^*}$.

1.6 User and Video Profilers

In this section, we detail the functionality of the UP and VP modules. We first describe different patterns in user behavior and tastes, and then propose the user profiler algorithm. The VP module is also briefly explained.

1.6.1 Profiling User Behavior

The user profiler runs on the client device and characterizes each user through (i) the video request probability at each time interval, and (ii) the distribution of video types preferred by the user.

User Viewing Pattern and Taste: Depending on their lifestyles, different users have different time preferences for watching videos. For example, some users prefer watching videos on weekends rather than on weekdays, while others watch more in the evening after working hours than in the mornings. The taste in content of the users can also be different. For example, some users watch sports videos more often than movie trailers, while some others watch more news clips than music videos. Such preferences in user behavior can lead to well-defined patterns, both in terms of the viewing times and the types of the videos being watched.

The job of the user profiler is to estimate these temporal viewing patterns and video type preferences for each user. The UP module does this based on the user's past video request records, spanning either the previous billing cycle, or the entire history. In this work, we consider requests from the last billing cycle.

Computing Video Request Probability: In each time interval t, there is a certain probability p_t that the user requests a video. This request probability can either vary with each interval or be constant. As a first attempt, we compute the *average* request probability per interval, and set p_t for each interval equal to this average. The average request probability is computed by summing the number of requests in the previous billing cycle and dividing by the number of periods T. The time interval should be set small enough so that the average request probability is less than 1, but not so small as to inhibit the computation of (1.3).

There are several alternative approaches, including fitting distributions and prediction-based techniques. The arrival rate of videos might follow a particular known distri-

bution (e.g., Poisson), in which case the probability of an arrival can be computed directly from the distribution itself. Alternatively, one can use more sophisticated time series analysis techniques. For example, at the beginning of the billing cycle, one can predict the sequence of future viewing times in the upcoming billing cycle, then compute the average request probability by adding up the predicted number of requests, and finally dividing that by the number of intervals. One can also design online algorithms, such as predicting the sequence of viewing times for intervals $t+1, t+2, \ldots, T$, while at interval t, and updating the predictions when a new video request arrives. Such alternatives trade off accuracy versus computation need.

We have developed one such online algorithm based on "triple exponential smoothing" [23]. However, here we will employ the simple averaging technique previously mentioned. The resulting computation requires less memory and power, and can be performed easily on a resource-constrained (in terms of battery and memory) client device. Our goal is not necessarily to develop the best user profiler, but to find a method that works well in the system as a whole. To establish this, we run trace-driven simulations in Section 1.7 to compare the performance of QAVA when the UP module uses the average request probability, to a scenario when the UP module has perfect knowledge of future arrivals. We find that our technique, while simple, achieves close to optimal performance (more than 95% on average).

Algorithm 1.2

User Profiling Algorithm

Input: Time stamps and utility--cost vectors $(\mathbf{u}_t, \mathbf{c}_t)$ of each video request in the previous billing cycle.
Output: Video request probability p_t, $\forall t$, and the video type joint probability distribution $P(\mathbf{u}, \mathbf{c})$.

1. Count the number of requests n_r, and the number of time intervals T in the last billing cycle.
2. Compute average request probability as $\bar{p} = n_r/T$, and set $p_t = \bar{p}$, $\forall\, t$.
3. Count the number of times each $(\mathbf{u}_t, \mathbf{c}_t)$ pair appears in the past; denote this count by $n_{(\mathbf{u}_t, \mathbf{c}_t)}$.
4. Construct the joint probability distribution by computing the individual probabilities as: $p(\mathbf{u}_t, \mathbf{c}_t) = n_{(\mathbf{u}_t, \mathbf{c}_t)} / \sum_{(\mathbf{u}'_t, \mathbf{c}'_t)} n_{(\mathbf{u}'_t, \mathbf{c}'_t)}$

Computing Video Type Distribution: The joint probability distribution $P(\mathbf{u}, \mathbf{c})$ reflects a user's preference for watching different types of videos. For example, a user who watches a lot of sports videos (which are not-so-compressible) will have a different distribution from a user who watches a lot of talk shows (more compressible). This video type distribution can remain the same over the length of a billing cycle, or can be time-dependent, reflecting, for instance, the fact that a user watches more sports videos at night and more news clips in the morning. As a first-order approximation, we assume that the distribution does not change with time. The distribution is computed once at the beginning of a billing cycle based on the video requests in the last billing cycle.

Our method is as follows: Each video request arriving at time interval t in the previous billing cycle has a $(\mathbf{u}_t, \mathbf{c}_t)$ pair associated with it. The probability distribution is calculated by counting the frequency of each $(\mathbf{u}_t, \mathbf{c}_t)$ pair from the last billing cycle, and then normalizing appropriately to form a probability distribution. Since the utility and cost are continuous variables, they can be binned for greater computational efficiency; however, in our dataset we find this optimization unnecessary. Through simulation, we show that this estimate performs very well, compared to the ideal scenario when the distribution of the requested videos is perfectly known ahead of time. Our user profiling algorithms are summarized in Algorithm 1.2.

1.6.2 Profiling Video Cost and Utility

The purpose of the video profiler running on the VP module is to estimate the utility and cost for all the bitrate versions of all videos stored on the content provider's server. There are many estimation techniques for computing the quality of a video. One standard objective metric is the PSNR, which we employ. The PSNR is a well-known objective metric for measuring video quality. A video typically comprises a sequence of images, and the PSNR for each image is defined as a function of the mean square error (MSE) between the original and compressed image. Mathematically, it is expressed in the logarithmic unit of decibel (dB) as PSNR $= 10\log_{10}(Q^2/D)$, where D is the pixel-wise MSE between the original and reconstructed image, and Q is the maximum pixel value (usually 255). We compute the video PSNR as the PSNR averaged over all images in the sequence. Typical PSNR values for lossy video compression are between 20 and 30 dB, where higher is better. To account for the duration of the video, we set the utility equal to the PSNR $\times$ the video duration. An example utility-cost curve is shown in Figure 1.3.

There exist other, potentially more accurate, metrics of video utility (e.g., Mean Opinion Scores [24] or MOS), as well as means of calculating subjective metrics from objective measurements [25, 26, 27]. However, we choose PSNR as it can be easily computed using a mathematical formula (in constrast to MOS that requires time consuming human experiments) and is very well known to the multimedia community.

Measuring the cost of a video in bytes naturally follows from the fact that the data quota is measured in bytes. The video profiler calculates the utility and the cost in MB for all the videos only once. These utility and cost vectors are stored alongside the videos on the content provider's server.

1.7 Performance Evaluation

We evaluate the performance of QAVA stream selection by comparing it with three alternatives: (i) a hindsight-based offline, optimal algorithm that knows all the video requests in a billing cycle ahead of time; (ii) a worst-case online algorithm; and (iii) a naïve method (used by, for example, Netflix). We also explore the sensitivity of QAVA to user profiler prediction errors.

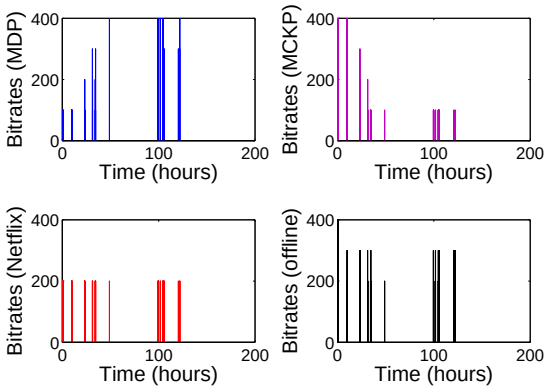


Figure 1.5 Bit rate selection by different algorithms for a single user. The MDP and MCKP algorithms choose different bitrates over time, while Netflix chooses a constant bitrate and the offline algorithm chooses the optimal bitrates.

1.7.1 Experimental Setup

Our simulations are based on the public-domain traces of 2 weeks of YouTube video requests from a wireless campus network [28]. The data comprises 16,337 users making a total of 611,968 requests over 14 days. YouTube is the largest source of video traffic, so the dataset captures a major portion of video viewing behavior [2]. The first week of trace data is used to train both video and user profilers. The second week emulates a billing cycle where the stream selector is run for each user's requests.

Each video is encoded in H.264/AVC at 100, 200, 300, 400, and 500 Kbps. The stream selector chooses one bitrate from the first four choices. The 500 Kbps version is treated as a reference for computing the PSNR of the other bitrates. The cost of each video is its size in MB. We set the user's quota at the halfway point between the minimum data usage (always selecting 100 Kbps) and the maximum data usage (always selecting 400 Kbps), and also sweep across quotas when appropriate. The period length is set to 30 minutes, since we experimentally find that varying the period length does not greatly change system performance.

One limitation of this evaluation is that not all videos were available from YouTube at the time of this study. In the training phase, missing videos are not included while generating the video type probability distribution. In the test phase, the utility–cost curves of missing videos are sampled from the video type distribution of the training phase. This gives an advantage to our algorithm, because the probability distribution of the training phase is similar to that of the test phase. We also examine the effect of misestimation of the distribution.

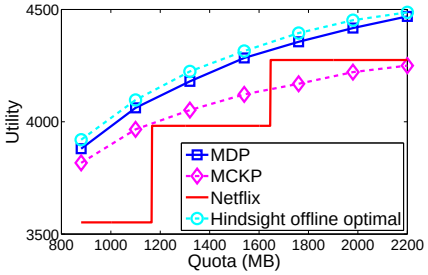


Figure 1.6 Quality–cost tradeoff for a single user with different quota and fixed video requests. MDP obtains close to optimal utility, while MCKP and Netflix perform sub-optimally.

1.7.2 Comparing Stream Selection Algorithms

We first evaluate the offline algorithm that solves problem (1.1) with the knowledge of all future video requests. It achieves the best possible performance and is treated as the benchmark against which we compare the performance of the online algorithms. We call this the *hindsight offline optimal* algorithm.

Zhou *et al.* present an online algorithm to solve (1.1) with a worst-case performance guarantee regardless of the sequence of video requests [29]. We call this the online *MCKP* algorithm and chose this to compare with our MDP algorithm because it optimizes for the *worst-case* performance, while our algorithm optimizes for *expected* performance. The MCKP algorithm, however, uses less information than our MDP algorithm, needing only the maximum and minimum utility-to-cost ratio across all requested videos, and an estimate of the sum data of the smallest bitrates. The MCKP algorithm does not use prediction or time deadlines, but requires only the quota.

The second online algorithm we compare with is the naïve solution currently used by Netflix. Netflix allows subscribers to select a default streaming bitrate. We assume that a Netflix user chooses one bitrate for the entire billing cycle, and is also intelligent enough to presciently choose the maximum bitrate that fits all videos in the quota. *Clearly, this algorithm is an ideal algorithm and not suitable for practical use, because it assumes advance knowledge of the number of videos to be watched.* Comparison with this Netflix method allows us to evaluate how our MDP algorithm performs against an existing practical solution.

1.7.3 Single User Examples

To understand the operation of the stream selector, we first run the algorithm for a single user with a target quota of 1426 MB (see Figure 1.7.1). The video requests arrive in bursts, and each algorithm selects bitrates. The Netflix method always chooses 200 Kbps. The MDP algorithm has foresight and thus chooses lower bitrates in the beginning of the billing cycle, knowing to save for later. The MCKP algorithm does not use time deadlines, only the remaining quota, and so it chooses high bitrates

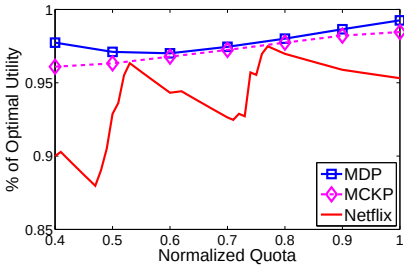


Figure 1.7 Quality–cost tradeoff averaged and normalized over multiple users. The MDP algorithm achieves nearly optimal performance, with the MCKP algorithm close behind. Both algorithms outperform the naïve Netflix method.

in the beginning before cutting back as it starts depleting the quota. The offline optimal algorithm chooses a variety of bitrates over time.

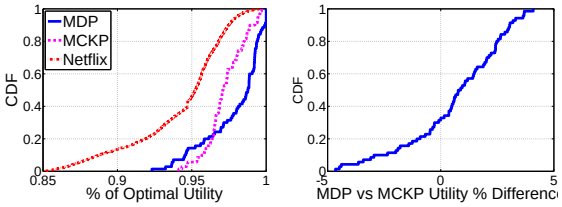
We also sweep across different monthly quotas and measure the utility obtained by each algorithm (see Figure 1.6). The offline algorithm performs the best, with MDP and MCKP close behind. The Netflix method exhibits a staircase-like shape, because as the quota increases, the default user-selected bitrate can increase. In all cases, the algorithms use less data than the target quota.

1.7.4 Multi-User Stream Selection

Average Performance: We now present the evaluation results of the MDP algorithm for multiple users. Each trial takes as input a fixed set of video requests and a quota, and computes the utility obtained by each algorithm. Some input combinations achieve higher utility than others. In order to fairly compare across multiple trials, we normalize the utility across different users by measuring the utility of the online algorithm as a fraction of the offline optimal utility. To normalize the quota, we measure data as a fraction of the total data if the 400 Kbps bitrate were always selected.

Figure 1.7 shows the average utility across 10 different users. We observe that, on average, the MDP algorithm performs better than the MCKP algorithm. The Netflix method resembles a staircase function as in the single-user case, and obtains especially low utility for low quotas. This is arguably the most important scenario: When the user's quota is small compared to the number of videos she wishes to watch. For these low quotas, the MDP has a definite advantage over MCKP, which in turn outperforms the Netflix method.

Performance Variability: To examine the utility distribution, and not just the average, we plot their cumulative distribution functions (CDFs) in Figure 1.8(a) across multiple quotas and users. The ideal result is a step function at 1, indicating 100% of the trials result in optimal utility. We see the Netflix method performs the worst, obtaining, for example, less than 95% utility 50% of the time. The MCKP curve is steeper, indicating it has less performance variation, which makes intuitive sense as



(a) (b)

Figure 1.8 (a) CDF of the utility achieved by MDP, MCKP, and Netflix algorithms. (b) Distribution of MDP performance improvement over MCKP.

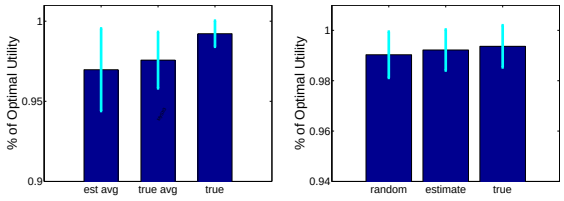
the algorithm optimizes for the worst-case. The MDP algorithm optimizes for the average-case performance but not the spread, and thus exhibits greater variation, but is closer to the ideal step function.

The MDP and MCKP algorithms are further compared in Figure 1.8(b), which shows the CDF of their percentage utility difference. If the MDP were always better than the MCKP, we would see a step function at 0. However, we observe that the MCKP sometimes outperforms the MDP algorithm. This may occur because the MCKP optimizes for worst-case performance, so when a user's behavior exhibits high variability, the MCKP is better able to handle the user's requests.

On the surface, these simulations seem to suggest the Netflix method performs reasonably well in general. It achieves above 85% of the optimal utility, suggesting that this naïve solution is acceptable for most users. However, the caveat is that our simulated Netflix method assumes perfect knowledge of the number of video requests in the billing cycle, so that the user knows how to correctly set the default bitrate given the quota. This is represented by the sharp jumps with increasing quota in Figure 1.6 and 1.7. If the user sets the default bitrate too high, she will overshoot her quota. If the user sets the default bitrate too low, she will obtain suboptimal utility. A main advantage of QAVA is that it automatically adjusts the bitrate, so the Netflix user does not need to estimate her usage to set her default bitrate, which might result in these over- or under-shooting problems.

1.7.5 Sensitivity to Prediction Error

It is important to examine how errors from the UP module affect the performance of the stream selection algorithm. There are two possible sources of error: video request probability and video type distribution. To test sensitivity to request probability error, we measure the utility obtained by the MDP algorithm when it uses (1) the estimated average arrival rate trained on historical data, (2) the true average arrival rate of the test data, and (3) the true request times. These results are averaged across multiple users and shown in Figure 1.9(a), with the error bars indicating standard deviation. We observe that the greater the information accuracy, the greater the average utility. The performance difference is quite small, which suggests that the MDP algorithm performs well independent of arrival probability accuracy. In addition to



(a) Arrival rate (b) Video type

Figure 1.9 Effect of user profiler prediction error.

the performance results shown in Fig. 1.9(a), the average difference between the true and estimated arrival rate (the input to the algorithm), is 8%. This suggests our user profiling technique is sufficient to capture variability of user behavioral patterns.

To analyze the sensitivity of the stream selector to video type prediction errors, we perform the following experiment. We calculate the true video type distribution of the test data, and also randomly generate a video type distribution. This random distribution has both random videos (drawn from the pool of requested videos from all users), and random probabilities. The utility obtained by the MDP algorithm using the random, estimated, and true distributions, averaged across all users, is shown in Figure 1.9(b) with the error bars indicating standard deviation. We find that the average utility increases only slightly as more accurate information is known, suggesting robustness of the MDP algorithm to video type distribution errors.

1.8 Conclusions

Two emerging trends of Internet applications, *video traffic becoming dominant* and *usage-based pricing becoming prevalent*, are at odds with each other. Current approaches in practice center on either decreasing video consumption through self-imposed warnings, or decreasing data consumption by lowering video quality, both of which degrade user quality of experience. Given this conflict, is there a way for users to stay within their monthly data plans (data quotas) without suffering a noticeable degradation in video quality? We proposed an online video adaptation mechanism, Quota Aware Video Adaptation (QAVA), that manages this tradeoff by leveraging the compressibility of videos and by predicting consumer usage behavior throughout a billing cycle. QAVA automatically selects the best bit rate to enable the consumer to stay under her data quota, while suffering minimal distortion. We developed the QAVA architecture and its main modules, including Stream Selection, User Profiling, and Video Profiling. Online algorithms were designed through dynamic programming and evaluated using real video request traces. Empirical results based on real video traces showed that QAVA performed better than existing approaches in literature and practical solutions. This suggests that QAVA can provide an effective solution to the dilemma of usage-based pricing of heavy video traffic.

REFERENCES

1. "Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update", 2011-2016.
2. "Global Interet Phenomena Report", *Sandvine*, 2012.
3. "Reliance 3G Plans & Pricing", <http://www.rcom.co.in/Rcom/personal/3G/HTML/PostpaidDataPlans.html>.
4. "Rogers Hi-speed Internet FAQ", <http://www.keepingpace.ca/faq.html>
5. "EU cracks down on "bill shock" roaming horror stories", *arstechnica*, http://arstechnica.com/tech-policy/2010/03/eu-cracks-down-on-bill-shock-roaming-horror-stories/
6. "FCC holds off on "bill shock" rule as industry plans alerts for reaching monthly limits", *Washington Post*, http://www.washingtonpost.com/blogs/post-tech/post/fcc-holds-off-on-bill-shock-rule-as-industry-plans-alerts-for-reaching-monthly-limits/2011/10/17/gIQAF0CbrL_blog.html
7. Onavo, <http://www.onavo.com/>.
8. *DataWiz*, <http://www.datami.com/>.
9. "Ice Cream Sandwich — Android Developer" http://developer.android.com/about/versions/android-4.0-highlights.html
10. Chang SF, Vetra A, "Video Adaptation: Concepts, Technologies, and Open Issues", in *Proc. IEEE*, 93(1):148-58, 2005.
11. Rejaie R, Handley M, Estrin D, "Quality adaptation for congestion controlled video playback over the Internet", *ACM SIGCOMM* 1999.
12. Liu J, Li B, Zhang Y, "An End-to-End Adaptation Protocol for Layered Video Multicast Using Optimal Rate Allocation", in *IEEE Trans. Mult.*, 6(1):87-102, 2004.
13. Liu X, Dobrian F, Milner H, Jiang J, Sekar V, Stoica I, Zhang H, "A Case fo a Coordinated Internet Video Control Plane", *ACM SIGCOMM*, 2012.
14. Liu C, Bouazizi I, Gabbouj M, "Rate Adaptation for Adaptive HTTP Streaming", *ACM MMSys*, 2011.
15. Akhshabi S, Began AC, Dovrolis C, "An Experimental Evaluation of Rate-Adaptation Algorithms", *MMSys*, 2011.
16. Schwarz H, Marpe D, Wiegand T, "Overview of the Scalable Video Coding Extension of the H.264/AVC Standard", in *IEEE Trans. Circuits Sys. Video Tech.*, 17(9), 1103-20, 2007.
17. MPEG-DASH, <http://dashpg.com/>.
18. Jan RH, Lin CP, Chern MS, "An optimization model for Web content adaptation", *Computer Networks* 50(7):953-65, 2006.
19. Bovik A, *The Essential Guide to Video Processing*, Elsevier 2009.

20. Kellerer H, Pferschy U, Pisinger D, *Knapsack Problems*, Springer 2004.
21. Puterman ML, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley 2005.
22. Papastavrou JD, Rajagopalan S, Kleywegt AJ, "The Dynamic and Stochastic Knapsack Problem with Deadlines", *Management Science*, 1996.
23. Winters PR, "Forecasting Sales by Exponentially Weighted Moving Averages", *Management Science* 6(3):324-42, 1960.
24. "Recommendation BT.500: Methodology for the subjective assessment of the quality of television pictures", *International Telecommunication Union*, 2012.
25. Wang Y, Schaar M, Chang S, Loui AC, "Classification-Based Multidimensional Adaptation Prediction for Scalable Video Coding Using Subjective Quality Evaluation", in *IEEE Trans. Circuits Sys. Video Tech.*, 15(10):1270-8, 2005.
26. Dobrian F, Sekar V, Awan A, Stoica I, Joseph DA, Ganjam A, Zhan J, Zhang H, "Understanding the Impact of Video Quality on User Engagement", *ACM SIGCOMM*, 2011.
27. Chen KT, Huang CY, Huang P, Lei CL, "Quantifying Skype User Satisfaction", *ACM SIGCOMM*, 2006.
28. Zink M, Suh K, Gu Y, Kurose J, "Watch Global Cache Local: YouTube Network Traces at a Campus Network - Measurements and Implications", *IEEE MMCN*, 2008.
29. Zhou Y, Chakrabarty D, Lukose R, "Budget Constrained Bidding in Keyword Auctions and Online Knapsack Problems", *WWW*, 2007.
30. Chen J, Ghosh A, Magutt J, Chiang M, "QAVA: Quota-Aware Video Adaptation", *ACM CoNEXT*, 2012.