Achieving Viewing Time Scalability in Mobile Video Streaming Using Scalable Video Coding

Cheng-Hsin Hsu*
Deutsche Telekom R&D Lab USA
5050 El Camino Real #221
Los Altos, CA 94022

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

ABSTRACT

We propose a general quality-power adaptation framework that controls the perceived video quality and the length of viewing time on battery-powered video receivers. The framework can be used for standalone video devices (e.g., DVD players and notebooks) as well as mobile receivers obtaining video signals from wireless networks (e.g., mobile TV and video streaming over WiMAX). Furthermore, the framework supports both live streams (e.g., live TV shows) and pre-encoded video streams (e.g., DVD movies). We present an adaptation algorithm for each mobile device to determine the optimal substream that can be received, decoded, and rendered to the user at the: (i) highest quality for a given viewing time, and (ii) longest viewing time for a given quality without exceeding the battery level constraint. We instantiate this framework and work out its details for mobile video broadcast networks. In particular, we propose a new video broadcast scheme that enables mobile video devices to efficiently adapt scalable video streams and achieve power saving proportional to the bit rates of the received streams. We implement the proposed framework in an actual mobile video streaming testbed and we conduct experiments using real video streams broadcast to mobile phones. These experiments show the practicality of the proposed framework and the possibility of achieving viewing time scalability. For example, on a mobile phone receiving and decoding the same video program, a viewing time in the range from 4 to 11 hours can be achieved by adaptively controlling the frame rate and visual quality of the video stream.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Modeling Techniques

General Terms

Design

*This work was done while the author was a PhD student at Simon Fraser University.

MMSys'10, February 22–23, 2010, Phoenix, Arizona, USA.

Keywords

Quality-of-service adaptation, power models, quality models

1. INTRODUCTION

Video streaming to mobile devices has become very popular, as modern mobile devices are powerful enough to render video contents that were only feasible to stationary workstations a few years ago. In fact, we have been witnessing a new usage pattern of mobile devices, which clearly indicates that more and more users consume video contents using their mobile devices [27]. Several studies, e.g., [22], reveal that unlike the CPU speed, memory size, and disk capacity, which achieved exponential growth in the past few decades, the battery technology has been lagging and only achieved a linear growth. Therefore, insufficient battery capacity imposes stringent energy constraints on mobile video streaming, which requires mobile device designers to take energy consumption as one of the main design considerations.

Since most traditional multimedia equipments have no energy constraints, users typically seek the optimal viewing experience in terms of resolution, frame rate, and picture quality. On mobile devices, however, users *must* consider the battery lifetime as a new dimension of their view experience because it determines the maximum viewing time. We define the battery lifetime as the amount of time a user can continuously watch mobile video streaming without charging or replacing the battery. To illustrate the importance of battery lifetime, consider a user, Amy, who wants to watch a 30-min TV episode using her cellular phone that only has remaining battery capacity for watching the show for 25 min. Most current mobile devices can not *adapt* to the energy constraint, because video streams coded by traditional, nonscalable, coders must be received and decoded in their entirety. Consequently, Amy would watch the episode for 25 min, and then miss the (most important) ending, which significantly degrades her viewing experience and may drive her away from the mobile video streaming service. From Amy's point of view, finishing this episode, but at a slightly lower video quality is probably a much better experience. Nonetheless, such adaptation is not possible on current mobile devices.

In this paper, we study the problem of: (i) predicting battery lifetime of mobile devices for video streaming services, and (ii) allowing users to opt for longer battery lifetime by watching the video at a lower picture quality. Our goal is to design a systematic and intuitive method for users to decide on the *desired* perceived quality and viewing time. To achieve this, we leverage scalable video coders (SVCs) that

encode each video content into a single video stream with multiple layers [24]. Scalable coded streams can be transmitted and decoded at various bit rates, and this is done by simple manipulations that extract substreams of a few layers from the original stream. Each substream can be rendered at a lower perceived quality than the original (complete) stream. SVC coded streams are conventionally used to support heterogeneous devices in terms of communication bandwidth, display resolution, and CPU power. In this paper, we highlight another important benefit of SVC: enabling viewing time scalability. That is, we show that SVC can provide users with a *control knob* to prolong viewing time of videos on mobile devices by reducing the perceived video quality. We present *quantitative* models that map basic energy consumption and video bit rate to intuitive, and easy-to-understand, performance metrics such as the length of viewing time in hours and expected perceived quality in MOS (mean opinion score). More importantly, we propose an adaptation algorithm for each mobile device to determine the optimal substream that can be received, decoded, and rendered to the user at the: (i) highest quality for a given viewing time, and (ii) longest viewing time for a given quality without exceeding the battery level constraint.

The contributions of this paper are as follows.

- We propose a general quality-power adaptation framework, which can be used to systematically control the perceived video quality and the length of viewing time on battery-powered video receivers. The framework is general because it can be used for standalone receivers (e.g., DVD players and notebooks) as well as mobile receivers obtaining video signals from wireless networks (e.g., mobile TV and video streaming over WiMAX). Furthermore, the framework supports both live streaming (e.g., live TV shows) and pre-encoded streams (e.g., DVD movies).
- We propose a novel video broadcast scheme that efficiently enables mobile video devices to implement the adaptation framework. We analytically analyze this scheme and its power consumption. This broadcast scheme is of interest in its own right, as it allows heterogeneous mobile devices to receive different versions of the video stream and achieve power saving commensurate to the bit rates of the received streams. The scheme is designed to broadcast SVC video streams that efficiently support temporal and visual quality scalabilities at the same time.
- We propose a CPU power consumption model for decoding scalable video streams. Our model is an integration of previous models that were proposed in different contexts.
- We implement the proposed framework and models in a real mobile TV testbed and we conduct experiments using actual video streams broadcast to mobile phones. These experiments show the practicality of the proposed framework and the possibility of achieving viewing time scalability using scalably-coded video streams.

The rest of this paper is organized as follows. We provide an overview of the proposed framework in Sec. 2. We present the details of the proposed video broadcast scheme in Sec. 3. We study the communication power consumption in the same section. In Sec. 4, we describe the CPU power consumption and perceived quality models. We give our quality-power adaptation algorithm in Sec. 5. We present experimental results in Sec. 6. We summarize the related work in the literature in Sec. 7, and we conclude the paper in Sec. 8.

Table 1: List of symbols used in the paper.

Symbol	Description
l	viewing time
q	perceived quality
t	frame rate
δ	quantization step
$y(\cdot)$	consumed CPU cycles per second
$p(\cdot)$	system power consumption
$p_n(\cdot)$	communication consumption
$p_c(\cdot)$	CPU power consumption
$p_b(\cdot)$	background power consumption
ω	current battery level
v	battery voltage
T_o	overhead duration in mobile TV
γ	power saving of comm. networks
R	mobile TV network bandwidth
r	bit rate of each coded stream
l_t^δ	layer for frame rate t and quant. step δ
r_t^δ	bit rate of layer l_t^δ
$\boldsymbol{d}_t^\delta$	dependent layers of layer l_t^δ
S	number of TV channels
b	burst size of the base layer
$a(\cdot)$	bit rate of a substream
λ	communication power consumption
ψ	CPU efficiency factor
$q(\cdot)$	perceived video quality of a substream

2. OVERVIEW

In this section, we present an overview of the proposed quality-power adaptation framework as well as the quality and power models needed for the framework. For quick reference, we list all symbols used in the paper in Table 1, where bold symbols denote sets, e.g., $\boldsymbol{t}$ denotes the set of frame rates while t is a specific frame rate.

2.1 Quality-Power Adaptation Framework

We propose an adaptation framework, which gives users with a *control knob* to prolong viewing time of videos on mobile devices by reducing the video quality. The framework uses scalable video coders to encode each video into a scalable stream that can be decoded at various bit rates. That is, several substreams can be extracted from the scalable stream, where each substream has a lower bit rate, incurs a lower decoding complexity, and is rendered at a lower perceived video quality than those of the complete (original) stream. Hence, receiving, decoding, and rendering a substream at a lower video quality consumes less energy, and prolongs battery lifetime and viewing time.

The goal of our framework is to take users' desired viewing time (or perceived quality) as input, and systematically compute the optimal video substream to receive and render so that the desired viewing time (or perceived quality) is met without violating the given energy constraint. Notice

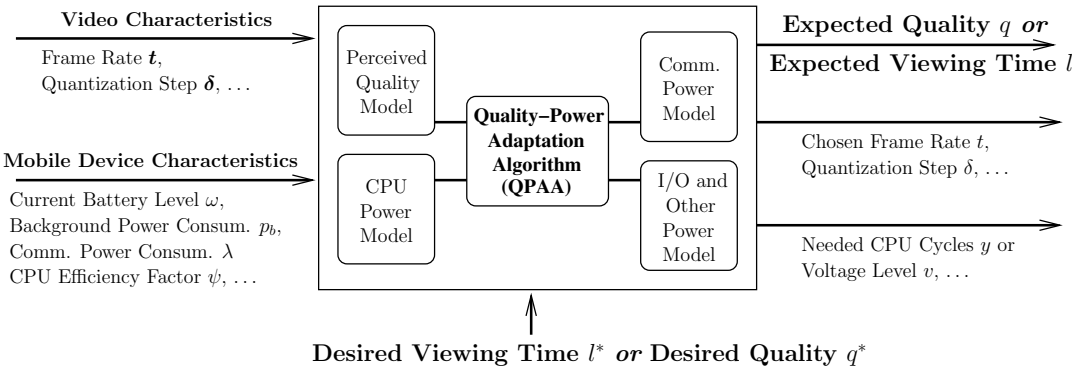


Figure 1: The proposed adaptation framework.

that the framework takes one desired performance metric, either viewing time *or* perceived quality and computes the expected value of the other metric. With this framework, Amy, in the illustrative example in Sec. 1, can specify the desired viewing time as 30 mins, and would not miss the ending of the TV episode, which is currently not possible on mobile devices. Fig. 1 illustrates the proposed framework. In the core of the framework is the Quality-Power Adaptation Algorithm (QPAA), which collaborates with four system models in order to produce the desired video adaptation strategy.

The adaptation framework takes several inputs, which we categorize into two classes: (i) video characteristics that describe the structure of coded streams, and (ii) mobile device characteristics that describe the conditions and efficiency of mobile devices. Examples of the inputs in the first class include supported frame rates $\boldsymbol{t}$ and quantization steps $\boldsymbol{\delta}$, which can be specified at encoding time or extracted from pre-encoded streams. Examples of mobile device characteristics include current battery level ω, background power consumption p_b, communication power consumption λ, and CPU efficiency factor ψ, which are device dependent and can either be directly measured or inferred by some experiments. The adaptation framework generates the adaptation strategy that consists of two parts: (i) the optimal version of substream to render specified by the chosen frame rate t as well as the quantization step δ, which determines the amount of *consumed* resources, and (ii) the output to control the amount of *supplied* resources, which include CPU voltage level v and available CPU cycles y. Applying these outputs to proper components leads to optimal stream adaptation.

2.2 Power and Quality Models

We briefly introduce the models used in the adaptation framework; more details are given in later sections.

The first model is the perceived quality model, which maps a given substream to its perceived video quality. The other three models specify the total power consumption of a mobile device, and they are: CPU, communication, and I/O and background models. The CPU power consumption p_c is the power consumed for *decoding* video streams. More CPU cycles in general lead to higher power consumption, and thus we write $p_c(y)$ as a function of y, which is the average number of CPU cycles per second used by video decoders. The communication power consumption p_n is the energy consumed for *receiving* video streams over communication networks. We consider mobile devices that can put their network interfaces into sleep for a fraction of time, and we define the power saving γ as the fraction of the time a network interface is in sleep mode over the total time. Since higher power saving γ means lower communication power consumption, we write $p_n(\gamma)$ as a function of γ.

The third part for the system power consumption is the I/O and background. The I/O power model calculates the power consumption of reading a given substream from a DVD, CD, flash memory, or any other storage medium, which is the case for standalone mobile video viewing systems. In this paper, we focus on mobile video streaming over communication networks, and thus we exclude the I/O power consumption from the discussion. The background power consumption p_b accounts for the power consumed in functions other than receiving and decoding video streams. For example, the power consumed for display backlights and for monitoring the control channels in cellular networks are part of the background power consumption. We model p_b as a constant. This is reasonable because rendering mobile video would occupy the small displays on mobile devices, which prevents users (and thus mobile devices) from multitasking. With these notations, we write the system power consumption as:

$$p(y, \gamma) = p_c(y) + p_n(\gamma) + p_b. \tag{1}$$

We give details on the functions $p_n(\gamma)$ and $p_c(y)$ in Secs. 3 and 4.1, respectively

Last, we map the system power consumption into viewing time l as follows. The battery capacity is often measured in mAh (milliampere-hour). Thus ω mAh means that the battery can sustain z mA for ω/z hrs. More and more mobile devices are equipped with built-in profiling capability to determine ω on-the-fly. We write the viewing time l as:

$$l = \frac{\omega v}{p(y, \gamma)}, \tag{2}$$

where v is the battery voltage.

3. COMMUNICATION POWER MODEL

Mobile devices can receive video streams over different types of wireless networks, including dedicated video broadcast networks, wireless LANs, WiMAX, and 3G cellular networks. Clearly, the power consumed by mobile devices to receive videos will depend on the operation of the specific network as well as the characteristics of the mobile devices.

In this paper, we focus on dedicated video broadcast networks (also known as Mobile TV), which concurrently offer video services to many users. We propose a new broadcast scheme that enables the adaptation of scalable video streams on mobile devices. Then, we develop the communication power model based on this scheme. Our analysis and models can easily be extended to support mobile video streaming over different wireless networks such as WiMAX.

3.1 Mobile Video Broadcast Networks

There are several standards proposed for for mobile video broadcast networks, such as MediaFLO (Forward Link Only technology) [11] and DVB-H (Digital Video Broadcast Handheld) [18]. MediaFLO is developed by Qualcomm and the FLO forum [10], and some details of its design are not public. DVB-H is an open international standard [6], and DVB-H networks have already been deployed in many countries around the world [4].

In a mobile video broadcast network, a base station broadcasts multiple video streams (alternately referred to as TV channels) in *bursts* with bit rates much higher than the actual encoding rates of the video streams. Thus, mobile devices can receive a burst of traffic and then turn off their network interfaces until the next burst in order to conserve energy. The next burst time is computed by the base station and included in the header fields of the current burst. This is called *time slicing* [18] and it is required in several broadcast standards such as DVB-H. Mobile devices must turn on their network interfaces slightly before the burst time, because it takes some time to wake up and synchronize the circuitry before they can start receiving data. This time is called the overhead duration and is denoted by T_o. With current technology, T_o is in the range of 50—250 msec [7, 18], which is nontrivial compared to burst lengths. We denote the power saving achieved by mobile devices because of time slicing as γ, which is the fraction of time the communication module is turned off to the total time. The rendering of video streams on mobile devices and the achieved power saving depend on the video broadcast scheme, which specifies the start time and the size of each burst of data for every video stream. The broadcast scheme cannot have burst collisions, which happen when two or more bursts have nonempty intersection in time. Moreover, the scheme must reserve enough burst time for each stream, so that all video data can be delivered on time.

The energy saving achieved by time slicing has been studied in the literature [7,13–16,30]. The authors of [30] and [7] estimate the effectiveness of time slicing on energy saving for any given broadcast scheme, but they do not construct broadcast schemes. In contrast, our previous works present algorithms for mobile TV base stations to compute optimal broadcast schemes to save energy using time slicing [13–16]. We considered homogeneous mobile devices, and proposed optimal/near-optimal broadcast schemes to send multiple nonscalable video streams to them [13, 15]. These broadcast schemes are not suitable for mobile devices with heterogeneous resources such as screen resolution, decoder capability, and battery level, which is the main focus of the current paper. In [14, 16], we considered heterogeneous mobile devices, and demonstrated that existing broadcast schemes do *not* efficiently support scalable video streams, and can lead to wasting the energy of mobile devices. We also proposed broadcast schemes for scalable video streams. However, our

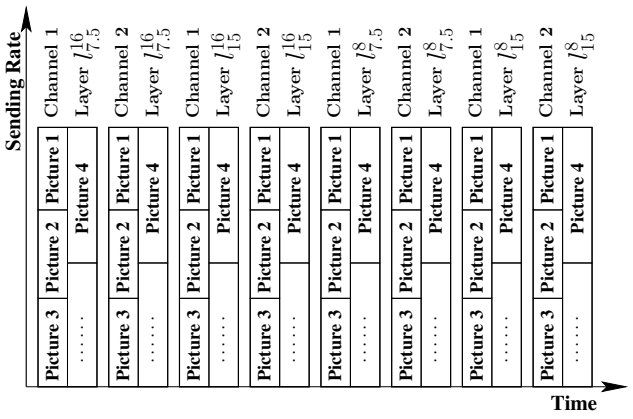


Figure 2: The proposed broadcast scheme.

broadcast schemes in [14, 16] have two limitations, which make them not suitable for the quality-power adaptation framework in the current paper. The first limitation is that these broadcast schemes allow video streams to only support one scalability mode: either temporal *or* visual quality, but not both at the same time. This prerequisite prevents video broadcast networks from supporting mobile devices with multiple scalability dimensions. The second limitation is that video streams are assumed to have simple, linear dependency model among layers. This limits inter-dependency among layers, and can result in coding inefficiency. In the current paper, we propose a new, more general, broadcast scheme that enables different mobile devices to adapt scalable streams along both the temporal and quality scalability dimensions at the same time. The new scheme, denoted by Flexible Mobile Video Broadcast (FMVB), also allows layers to be encoded at different bit rates and does not assume that layers are linearly cumulative. Thus, FMVB offers substantial flexibility to base stations as well as to mobile devices that was not possible in our previous schemes in [13–16]. More importantly, FMVB is more suitable to modern scalable video coders, such as H.264/SVC [24], which support multiple scalability modes, and complex inter-layer dependency for higher coding efficiency.

3.2 Flexible Mobile Video Broadcast (FMVB)

We now present the FMVB scheme. We consider a quite general scalable stream that supports both temporal and quality scalability modes. Let $\boldsymbol{t}$ be the set of all supported frame rates and $\boldsymbol{\delta}$ be the set of all supported quantization steps. We use l_t^δ to represent the layer that supports frame rate t ($t \in \boldsymbol{t}$) and quantization step δ ($\delta \in \boldsymbol{\delta}$). We notice that scalable video coders exploit the data redundancy among layers in order to achieve higher coding efficiency, and thus there exist *inter-layer dependencies* among the layers of each scalable stream. To model this dependency, we define $\boldsymbol{d}_t^\delta$ as the set of dependent layers of l_t^δ, which also includes l_t^δ itself. $\boldsymbol{d}_t^\delta$ consists of all layers that are needed to successfully decode l_t^δ. That is, $\boldsymbol{d}_t^\delta$ defines a decodable *substream* that can be extracted from the original (complete) stream. We let r be the bit rate of the original scalable stream with full quality, and R be the broadcast network bandwidth. Therefore, the number of TV channels that can be concurrently broadcast is $S = \lfloor R/r \rfloor$. We let r_t^δ be the bit rate of layer l_t^δ, and we have $r = \sum_{\delta \in \boldsymbol{\delta}} \sum_{t \in \boldsymbol{t}} r_t^\delta$. We note that

$\boldsymbol{\delta}$, $\boldsymbol{t}$, $\boldsymbol{d}_t^\delta$, r_t^δ, and r are coding parameters that are specified at encoding time. These parameters can also be extracted from pre-encoded video streams, and used by the proposed adaptation framework.

FMVB is illustrated in Fig. 2. It creates a different burst for each layer of every TV channel. As shown in Fig. 2, all IP packets in the left-most burst belong to the $l_{7.5}^{16}$ layer of TV channel 1, while IP packets in the third burst belong to layer l_{15}^{16} of TV channel 1. Since each burst consists of IP packets from the same layer of the same TV channel, mobile devices know which layer those IP packets are in, even before receiving the burst. This frees mobile devices from opening the network interfaces and inspecting IP packets for substream extractions, and results in *higher* energy saving. The FMVB works on recurring time windows, where each window consists of a burst of each layer of every TV channel. We let b be the burst size of the base layer, which is layer $l_{t_{\min}}^{\delta_{\max}}$, where $t_{\min}$ is the minimal frame rate and $\delta_{\max}$ is the maximal quantization step supported by the coded stream. Then the burst size of layer l_t^δ, where $t \in \boldsymbol{t}$ and $\delta \in \boldsymbol{\delta}$ is given as $b(r_t^\delta / r_{t_{\max}}^{\delta_{\min}})$, and the recurring window length is $\frac{br/r_{t_{\max}}^{\delta_{\min}}}{R} S = \frac{brS}{r_{t_{\max}}^{\delta_{\min}} R}$ sec. To arrange the broadcast time of each layer l_t^δ, where $t \in \boldsymbol{t}$ and $\delta \in \boldsymbol{\delta}$, we enumerate all layers and denote them as $\bar{l}_1, \bar{l}_2, \ldots, \bar{l}_E$, where $E = |\boldsymbol{\delta}| \times |\boldsymbol{t}|$. In addition, we use $\bar{r}_e$ to denote the layer bit rate of layer $\bar{l}_e$, where $e = 1, 2, \ldots, E$. We mention that this enumeration can be *arbitrarily* done, but the mapping between l_t^δ and $\bar{l}_e$ should be fixed. The FMVB scheme allocates a layer $\bar{l}_e$ burst for each TV channel s, where $s = 1, 2, \ldots, S$, in a round-robin fashion.

Mathematically, the FMVB scheme allocates a burst of size $b(\bar{r}_e / r_{t_{\max}}^{\delta_{\min}})$ at time:

$$\frac{b\sum_{i=1}^{e-1} \bar{r}_i / r_{t_{\max}}^{\delta_{\min}}}{R} S + \frac{b\bar{r}_e / r_{t_{\max}}^{\delta_{\min}}}{R}(s-1), \tag{3}$$

to layer e ($e = 1, 2, \ldots, E$) of TV channel s ($s = 1, 2, \ldots, S$).

We notice that the frame rates $\boldsymbol{t}$ and quantization steps $\boldsymbol{\delta}$ specify the scalable stream structure, and are inputs to scalable video coders. Various frame rates and quantization steps can be used to support heterogeneous mobile devices. In the rest of this paper, we consider several typical combinations of the frame rate and the quantization step: we let t be 3.75, 7.5, 15, and 30 fps, and δ be 16, 40, 64, and 104. Among the considered δ and t values, we let $\delta_{\min}$ be the smallest quantization step, and $t_{\max}$ be the highest frame rate. Note that the substream with quantization step $\delta_{\min}$ and frame rate $t_{\max}$ is the original video stream and should result in the highest possible perceived quality, which is denoted by $q_{\max}$. Similar frame rates and quantization steps are also employed in several recent works [20, 28].

3.3 Analysis and Power Model

We next study the relationship between the FMVB scheme and communication power consumption. Our goal is to build a model so that each mobile device knows the expected communication power consumption level of every potential substream. In the next theorem, we derive the relative power saving in terms of the fraction of time mobile devices can turn off their network interfaces. We also show that the FMVB scheme produces feasible, valid broadcast schemes.

THEOREM 1. *The FMVB scheme (specified by Eq. (3)) specifies a feasible broadcast scheme for a recurring window of size* $\left(brS\right)/\left(r_{t_{\max}}^{\delta_{\min}} R\right)$ *sec, where (i) no two bursts overlap with each other, and (ii) bursts are long enough to send data for all mobile devices to playout until the next burst. Furthermore, the energy saving achieved by mobile devices that render a substream with frame rate t and quantization step δ is given by:*

$$\gamma_t^\delta = 1 - \frac{\sum_{l_i^j \in \boldsymbol{d}_t^\delta} r_i^j}{rS} - \frac{\left|\boldsymbol{d}_t^\delta\right| T_o r_{t_{\max}}^{\delta_{\min}} R}{brS}, \tag{4}$$

where $t \in \boldsymbol{t}$ and $\delta \in \boldsymbol{\delta}$.

PROOF. First, sending a burst of $b\bar{r}_e / r_{t_{\max}}^{\delta_{\min}}$ kb takes time $\frac{b\bar{r}_e / r_{t_{\max}}^{\delta_{\min}}}{R}$. Thus, based on the definition of the broadcast scheme in Eq. (3), the scheme has no overlapping bursts. This is because the start time of each burst is ensured to be greater than the end time of its predecessor. Second, because the recurring window size is $\frac{brS}{r_{t_{\max}}^{\delta_{\min}} R}$ and $S = \lfloor R/r \rfloor < R/r$, the required amount of data of any layer e ($1 \le e \le E$) for smooth playout is:

$$\frac{brS}{r_{t_{\max}}^{\delta_{\min}} R} \times \bar{r}_e \le \frac{br}{r_{t_{\max}}^{\delta_{\min}} R} \times \frac{R}{r} \times \bar{r}_e = b\frac{\bar{r}_e}{r_{t_{\max}}^{\delta_{\min}}},$$

where $\bar{r}_e$ is the layer bit rate. This inequality shows that the reserved time period is long enough to carry the video data.

For power saving, observe that mobile devices that render a video with frame rate t and quantization step δ receive at bit rate $\sum_{l_i^j \in \boldsymbol{d}_t^\delta} r_i^j$, which is the aggregate bit rate of all dependent layers of layer l_t^δ, and the video data are delivered in $\left|\boldsymbol{d}_t^\delta\right|$ bursts within each recurring window of $\frac{brS}{r_{t_{\max}}^{\delta_{\min}} R}$ sec. Therefore, the power saving of mobile devices that render at frame rate t and quantization step δ is given as:

$$\gamma_t^\delta = 1 - \frac{b\sum_{l_i^j \in \boldsymbol{d}_t^\delta} \frac{r_i^j}{r_{t_{\max}}^{\delta_{\min}}}}{R} + \left|\boldsymbol{d}_t^\delta\right| T_o}{\frac{brs}{r_{t_{\max}}^{\delta_{\min}} R}}.$$

In this equation, the first term of the numerator accounts for the time to receive actual video data, the second term of the numerator represents the total overhead durations, and the denominator is the recurring window size. Rearranging this equation yields Eq. (4). □

This theorem gives the power saving γ in mobile video broadcast networks. Notice that the power saving decreases as mobile devices receive more layers, because of the summation in the second term in the right hand side of Eq. (4). That is, our FMVB scheme allows differentiation in power saving for heterogeneous mobile devices, which is an important property for the quality-power adaptation framework. This differentiation is not possible with other broadcast schemes, as mentioned in Sec. 3.1.

To compute γ from Eq. (4), we need to know the layer bit rate $\bar{r}_e$ for all layers $e = 1, 2, \ldots, E$. The layer bit rate can be specified at encoding time, extracted from pre-encoded streams, or predicted using the rate model proposed in [28], which writes the bit rate of a substream with quantization

step δ and frame rate t as:

$$a(\delta, t) = r(\delta/\delta_{\min})^{-\phi}(t/t_{\max})^{\chi}, \quad (5)$$

where r is the bit rate of the complete stream, and ϕ and χ are model parameters. The authors of [28] encode several video sequences, and extract substreams with different δ and t values. For each video sequence, 16 substreams are considered, and the model parameters in Eq. (5) are estimated using curve fitting. For example, the model parameters for the Crew sequence are $\phi = 1.234$ and $\chi = 0.671$. Using these parameters in Eq. (5) enables us to predict the layer bit rates with different quantization steps and frame rates.

This rate model allows us to compute γ using Eq. (4). Finally, since the power saving γ is the fraction of time a device can turn off its mobile TV chip, the communication power consumption can be written as:

$$p_n(\gamma) = \lambda(1 - \gamma), \quad (6)$$

where λ mW is the energy consumption of the mobile TV chip, which is a model parameter.

4. CPU POWER AND PERCEIVED QUALITY MODELS

In this section, we first present the CPU power model, which is followed by the perceived quality model.

4.1 CPU Power Model

We develop the CPU power consumption model for decoding scalable video streams in two steps. First, we describe a complexity model that predicts the number of CPU cycles required to decode a given substream of a scalable video stream. Then, we map the number of required CPU cycles into actual CPU power consumption. We describe these two steps in the sequel.

There are several complexity models for video encoders in the literature, while there is not much done for video decoders. One reason is that traditional, nonscalable, video decoders have little rooms for reducing computational complexity: the whole video stream *must* be decoded for playout. This, however, is not true for scalable coded streams, where video decoders can render a *substream* of the original stream in order to save CPU cycles. A recent work [20] proposes several variations of a complexity model for scalable video decoders. We adopt one variation that supports temporal and visual quality scalabilities. The temporal scalability is achieved by controlling the frame rate. The visual quality scalability is typically achieved by using different quantization steps during video compression: smaller steps produce better quality.

The complexity model takes δ and t as inputs, and gives the number of CPU cycles per second as:

$$y(\delta, t) = \frac{t_{\max}}{G} M \big[\theta Y_I + (1-\theta) Y_P + (t/t_{\min} - 1) Y_B + t/t_{\min} \log_2(\delta_{\max}/\delta) Y_Q\big], \quad (7)$$

where $\delta_{\max}$ is the largest quantization step and $t_{\min}$ is the lowest frame rate, G is the GoP (group of picture) size, M represents the number of macroblocks per picture, θ is the fraction of I-frame among all key pictures, and Y_I, Y_P, and Y_B are the average macroblock decoding complexity of I-, P-, and B-frames, respectively, and Y_Q is the average macroblock decoding complexity at a quality enhancement layer. We notice that this complexity model only consists of four model parameters: Y_I, Y_P, Y_B, and Y_Q, while all other variables are inputs.

The authors of [20] consider several video sequences and use the Intel VTune analyzer to profile an H.264/SVC decoder with various combinations of δ and t. For each video sequence, they fit Eq. (7) against the profiling data in order to derive the model parameters. Based on their experiments, the model parameters for the Crew sequence are $Y_I = 120791$, $Y_P = 274102$, $Y_B = 130787$, and $Y_Q = 102062$ cycles. For concreteness of our discussion, we consider typical inputs for CIF (352x288) sequences: $G = 8$, $M = 396$, and $\theta = 0.25$. With these parameters and constants, the complexity model in Eq. (7) can predict the number of CPU cycles per second required by video decoders for any given δ and t.

Next, we translate the number of CPU cycles per second to the power consumption of the CPU. The energy scaling function has been implemented in most recent CPUs, which allows mobile devices to reduce the power consumption by running slower [12]. The power consumption of a CPU can be written as $v^2 \times f \times e$, where v is the voltage, f is the clock frequency, and e is the effective capacitance of the CPU. Moreover, previous studies show that the clock frequency f is linearly proportional to the voltage v [12]. Therefore, the power consumption of CPU is proportional to f^3. We assume that mobile devices implement the dynamic voltage scaling (DVS) mechanism, and can adjust the clock frequency to match the *demand* of CPU cycles ($f = y$). This ensures the jobs are done just on-time without having the CPU idling, and thus minimizes the CPU power consumption, which is given as:

$$p_c(y) = \psi y^3, \quad (8)$$

where ψ represents the CPU efficiency, and is a model parameter. Combining Eqs. (7) and (8), we can compute the CPU power consumption for any given δ and t. The model parameter ψ is device dependent.

4.2 Perceived Quality Model

Modern scalable video coders, such as H.264/SVC [24], are fairly flexible, and substreams in various resolutions, frame rates, and qualities, can be extracted from a single scalable stream. Therefore, defining video quality metrics for these substreams is challenging because most traditional video quality metrics, such as PSNR (peak signal-to-noise ratio), assume that the resolution and frame rate are fixed. In the past few years, several quality metrics for scalable streams have been proposed. Some of these works attempt to "patch" PSNR, while others are based on subjective user studies. We employ the perceived quality model proposed in [28] for two reasons. First, it is based on user studies, and uses MOS as the quality metric. MOS is more appropriate than pixel-difference based objective metrics like PSNR. For example, the PSNR of a 7.5 fps video stream and that of a 30 fps video stream are *not* comparable. Second, this quality model has only two model parameters and yet is fairly accurate [28]. We describe this perceived quality model in the following.

The quality model considers two scalability modes: quality and temporal, and it writes the video quality q as a function of the quantization step δ and the frame rate t. The video quality q in MOS ranges from 0 to 100. The quality

Quality-Power Adaptation Algorithm

1. //Input: $\boldsymbol{\delta}, \boldsymbol{t}$: possible substreams
1. //Input: l^*: desired watch time
1. //Output: δ, t: optimal substream for adaptation
1. //Output: q: expected quality
2. **foreach** substream $\hat{\delta} \in \boldsymbol{\delta}$ and $\hat{t} \in \boldsymbol{t}$
3. **compute** CPU power consumption $p_c(\hat{\delta}, \hat{t})$
3. using Eqs. (7) and (8)
4. **compute** substream bit rate $a(\hat{\delta}, \hat{t})$ using Eq. (5)
5. **compute** communication power consumption
5. $p_n(\hat{\delta}, \hat{t})$ with Eqs. (4) and (6)
6. **compute** power consumption $p(\hat{\delta}, \hat{t}) = p_c(\hat{\delta}, \hat{t}) +$
6. $p_n(\hat{\delta}, \hat{t}) + p_b$
7. **compute** viewing time $l(\hat{\delta}, \hat{t})$ using Eq. (2)
8. **if** $l(\hat{\delta}, \hat{t}) < l^*$ **continue** // violation
9. **compute** $q(\hat{\delta}, \hat{t})$ using Eq. (9)
10. **if** $q(\hat{\delta}, \hat{t}) > q$ // better?
11. **let** $q = q(\hat{\delta}, \hat{t}), \delta = \hat{\delta}, t = \hat{t}$
12. **end if**
13. **endfor**
14. **return** δ, t, q

Figure 3: The proposed adaptation algorithm.

model gives the perceived quality as:

$$q(\delta, t) = q_{\max} \frac{e^{-\alpha \frac{\delta}{\delta_{\min}}}}{e^{-\alpha}} \frac{1 - e^{-\beta \frac{t}{t_{\max}}}}{1 - e^{-\beta}}, \quad (9)$$

where $q_{\max}$ is constant and α and β are model parameters.

The authors of [28] derive the model parameters of several video sequences. For each video sequence, four δ values and four t values are considered, which lead to 16 substreams in total. The subjective test is repeated for all 16 substreams of each video sequence, which results in 16 samples for Eq. (9). Finally, the constant $q_{\max}$ and the model parameters α and β in Eq. (9) are estimated from these samples using least square error fitting. For example, the model parameters for the Crew sequence are $q_{\max} = 89$, $\alpha = 0.17$, and $\beta = 7.34$. Using these parameters, Eq. (9) enables us to predict the perceived video quality of each substream.

5. QPAA: THE PROPOSED ADAPTATION ALGORITHM

We present an algorithm that integrates all models presented in Secs. 3 and 4. The algorithm provides a systematic method for controlling the tradeoff between the perceived visual quality and energy consumed by mobile devices. Fig. 3 presents a high-level pseudo code of the proposed quality-power adaptation algorithm (QPAA). This algorithm takes desired watch time l^* as input and returns a substream with the highest quality q without violating energy constraint. The foreach loop between lines 3–13 considers all supported substreams, which are indicated by the set of possible quantization steps $\boldsymbol{\delta}$ and the set of possible frame rates $\boldsymbol{t}$. For each substream, the algorithm computes the power consumption in lines 3–6. It then computes the expected viewing time in line 7. Line 8 checks whether the expected viewing time $l(\hat{\delta}, \hat{t})$ satisfies the input l^*, and the algorithm moves to the next substream if $l(\hat{\delta}, \hat{t}) < l^*$. The algorithm computes the expected quality in line 9, and it searches for the optimal substream using the if statement in lines 10—12. Last, the optimal substream δ, t, and expected quality q are returned in line 14.

Notice that, as illustrated in Fig. 1, the QPAA algorithm also works in the other way around with some straightforward modifications in the pseudo code in Fig. 3. The modified QPAA takes the desired quality q^* as input, and returns the optimal substream and the expected viewing time l.

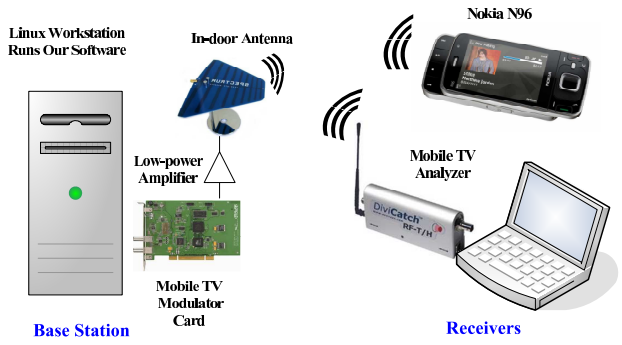


Figure 4: Setup of the testbed.

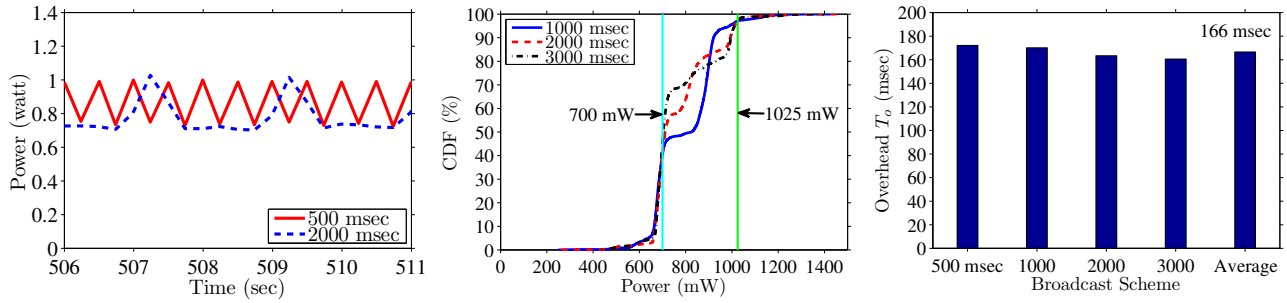
6. EXPERIMENTAL EVALUATION

In this section, we use a real mobile TV network to show the potential of viewing time scalability. More specifically, we try to answer the following question: how much viewing time can be prolonged by rendering a substream at a lower quality?

6.1 Testbed Setup

We have implemented a mobile TV testbed in our lab. This testbed provides us a realistic platform for analyzing the performance of mobile video broadcast networks. As illustrated in Fig. 4, the testbed has two parts: base station and receivers. We use a commodity Linux box as the base station, and have developed a software package on it to broadcast video streams. Most of the software components were developed by us and few libraries and drivers were leveraged from open source projects. We have installed a PCI modulator card in the base station, which supports the physical layer of the DVB-H protocol. Our software package implements the link layer of the DVB-H protocol and drives the modulator card to transmit DVB-H standard compliant signals via a power amplifier and an indoor antenna.

We use the Nokia N96 phone as the receiver, which is the most recent Nokia phone that supports mobile TV at the time of writing. This cellular phone allows us to assess the visual quality of videos, and serves as the target platform for parameter estimations detailed in Sec. 6.2. Our mobile TV testbed also consists of a DVB-H signal analyzer, which enables us to conduct detailed analysis on the mobile TV network. This analyzer is attached to a PC via a USB port and is supported by an analysis application. This analysis application records traffic streams as well as provides various statistics.



(a) Total power consumption (b) Total power consumption distribution (c) Overhead duration

Figure 5: Estimating the communication related parameters of the Nokia N96 phone.

6.2 Parameter Estimation and Results

There are several device dependent model parameters required by the QPAA algorithm. These include: (i) λ in Eq. (6), which is the power consumption of the mobile TV chip , (ii) T_o in Eq. (4), which is the overhead duration, (iii) p_b in Eq. (1), which is the background power consumption, and (iv) ψ in Eq. (8), which represent the power efficiency of the CPU. We explain how to derive them in the following.

Mobile TV Chip Power Consumption. We design several broadcast schemes with different parameters in order to infer the mobile TV chip power consumption λ. We capture a 10-minute news clip from a digital cable service. We encode the video using an H.264/AVC encoder at bit rate 450 kbps, and the audio using an eAAC+ encoder at 32 kbps. We make the base station broadcast the coded video stream, and we restart the stream once its end is reached. We broadcast the video stream for 3.5 hrs. We use an N96 phone to watch this video, and we measure its power consumption using a monitoring program called Juice, which has been shown to be fairly accurate and is comparable to external instruments [2]. After each 3.5-hr experiment, we fully charge the battery, and repeat the experiment by broadcasting the same video but with a different inter-burst time period. We consider five different broadcast schemes with inter-burst periods: 250, 500, 1000, 2000, and 3000 msec. We collect the power consumption of each broadcast scheme throughout the experiment.

We first plot two sample curves of the N96 phone power consumption in Fig. 5(a); curves for other schemes are similar. This figure confirms that the broadcast schemes with longer inter-burst time periods result in fewer power spikes, and thus lead to lower power consumption. For example, between broadcast times 508 and 510, there is a single power spike for 2000-msec broadcast scheme, but there are four power spikes for the 500-msec scheme. More importantly, the *height* of these spikes represents the power consumption of the mobile TV chip used by Nokia N96. This is because power consumed by other components is rather *constant*, e.g., the CPU decodes and displays up to 30 frames every second, which is much more often than the chosen inter-burst time periods. To derive the mobile TV chip power consumption, we compute the CDF (cumulative distribution function) of all power consumption measurements for each broadcast scheme. We then plot CDF curves of three sample broadcast schemes in Fig. 5(b). This figure shows that the power consumption measurements form *two* clusters at about 700 mW and 1025 mW, which represents the mobile TV chip *off* and *on* time, respectively. Hence, this experiment reveals that the mobile TV chip used by Nokia N96 consumes $\lambda = 325$ mW. Note that the inferred λ value is quite close to public white papers, e.g., a data sheet released by a popular DVB-H chip manufacturer indicates that modern DVB-H chips have a power consumption of about 400 mW [3]

Overhead Duration. Next, we use the above experimental results to derive the overhead duration T_o. We draw an observation from the experiment setup: the power consumption difference between any two of the broadcast schemes is completely attributed to the number of overhead durations T_o they impose. This is because we broadcast exactly the same video stream in all schemes, and the energy consumed for receiving, decoding, rendering, and displaying the video are always the same. Therefore, we can derive the T_o value using λ and the difference on power consumptions of two broadcast schemes. To illustrate, we consider the power consumption curves of the two sample schemes in Fig. 5(a), and we let p_1 and p_2 be the average power consumption of the 500-msec and the 2000-msec scheme, respectively. Clearly, the 500-msec scheme incurs three more T_o than the 2000-msec scheme in every 2-sec time period. Therefore, we can estimate the T_o as $\frac{2(p_1 - p_2)}{3\lambda}$. To systematically infer T_o, we pick the 250-msec scheme as the baseline, and derive T_o by comparing its power consumption against that of other schemes. We derive the T_o value from each broadcast scheme. We plot the results in Fig. 5(c). This figure shows that the T_o values inferred from different schemes are quite consistent, and they have an average of 166 msec, with a very small variance. Thus, we set $T_o = 166$ msec in our later experiments.

Background Power Consumption. We conduct the following experiment to infer the background power consumption. We configure the N96 phone to display photos in a slide show mode for 1.5 hrs, and we use Juice to monitor the power consumption. We chose the photo application to prevent the phone from dimming its backlight, which is one of the main sources of background power consumption. Fig. 6(a) illustrates the power consumption over the whole experiment. Using this figure, we compute the average power consumption as 290.38 mW, which is essentially the background power consumption p_b. This is because the slide show scheme is the same as the mobile video streaming