# Broadcasting Video Streams Encoded With Arbitrary Bit Rates in Energy-Constrained Mobile TV Networks

Cheng-Hsin Hsu, *Student Member, IEEE*, and Mohamed M. Hefeeda, *Member, IEEE*

***Abstract*—Mobile TV broadcast networks have received significant attention from the industry and academia, as they have already been deployed in several countries around the world and their expected market potential is huge. In such networks, a base station broadcasts TV channels in bursts with bit rates much higher than the encoding bit rates of the videos. This enables mobile receivers to receive a burst of traffic and then turn off their receiving circuits till the next burst to conserve energy. The base station needs to construct a transmission schedule for all bursts of different TV channels. Constructing optimal (in terms of energy saving) transmission schedules has been shown to be an NP-complete problem when the TV channels carry video streams encoded at arbitrary and variable bit rates. In this paper, we propose a near-optimal approximation algorithm to solve this problem. We prove the correctness of the proposed algorithm and derive its approximation factor. We also conduct extensive evaluation of our algorithm using implementation in a real mobile TV testbed as well as simulations. Our experimental and simulation results show that the proposed algorithm: 1) is practical and produces correct burst schedules; 2) achieves near-optimal energy saving for mobile devices; and 3) runs efficiently in real time and scales to large scheduling problems.**

***Index Terms*—Broadcast networks, burst transmission, Digital Vido Broadcast-Handheld (DVB-H), energy saving, mobile video streaming, time slicing.**

## I. Introduction

THE mobile TV service is expected to be the next killer application for mobile devices such as smart phones and mobile media players. It extends the viewing time of users and provides more business opportunities to content providers. Although mobile TV networks have already been deployed in several countries and are being tried in many others [1], there is still a large room for optimizing their performance from different angles. This paper addresses the problem of minimizing energy consumption for mobile devices while allowing content providers to broadcast diverse video content encoded at arbitrary different and variable bit rates. Minimizing energy consumption in mobile TV networks is critical for the success and wide adoption of mobile TV service because most mobile devices are battery-powered. In fact, popular mobile TV standards such as Digital Video Broadcast-Handheld (DVB-H) [2], [3] and MediaFLO (Forward Link Only technology) [4] *dictate* using energy-saving schemes to increase the viewing time on mobile devices. The typical energy-saving scheme is to broadcast TV channels in *bursts* at bit rates much higher than the encoding rates of the video streams. Mobile devices can then receive a burst of traffic and turn off their radio frequency (RF) circuits till the next burst. This is referred to as *time slicing*. To enable time slicing for mobile devices, a base station broadcasting multiple TV channels needs to schedule the transmission of bursts belonging to all TV channels. The burst scheduling algorithm must not result in receivers' buffer over- or underflow instances that cause playout glitches and degrade viewing experience for any TV channel.

Current burst scheduling approaches are simple heuristics. For example, the base station in Nokia Mobile Broadcast Solution (MBS) [5] can only take a system-wide interburst time period, which is *manually* specified by the network operator. That is, all TV channels, despite the encoding bit rates, have the same number of bursts in every scheduling time window, while the burst length of each TV channel may be different. Manually selecting a *feasible* interburst time for TV channels with different bit rates is challenging and error-prone, while the resulting burst schedules may not be optimal in terms of energy saving. This is because when the TV channels have different bit rates, the number of bursts that each channel requires as well as the size of each burst will be different in optimal schedules. Therefore, each TV channel should not only have *diverse* but also *dynamic* interburst periods, which are not supported by current base stations such as Nokia MBS.

Being restricted by the current base stations, many mobile TV deployments had to resort to encoding all TV channels at the same bit rate and simply stagger TV channels next to each other. For example, the trial mobile TV service in Paris broadcast 13 TV channels all encoded at 270 kbps [6]. Encoding all TV channels at the same bit rate is clearly inefficient and may yield huge quality variations among TV channels carrying different kinds of programs. For example, encoding a sports game requires a much higher bit rate than encoding a talk show. If we encode all TV channels at the same high bit rate, some channels may unnecessarily be allocated more bandwidth than they

Manuscript received February 24, 2009; revised July 01, 2009; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Hofmann. First published November 17, 2009; current version published June 16, 2010. This work is partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada. A preliminary version of this paper appears in INFOCOM 2009.

The authors are with the School of Computing Science, Simon Fraser University, Burnaby, BC V5A 1S6, Canada (e-mail: cha16@cs.sfu.ca; mhefeeda@cs.sfu.ca).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2009.2033058

1063-6692/$26.00 © 2009 IEEE

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

require, and this extra bandwidth yields only marginal or no visual quality improvement. Thus, the expensive wireless bandwidth of the broadcast network could be wasted. On the other hand, if we encode all TV channels at the same low or moderate bit rate, not all channels will have good visual quality, which is annoying to users. To the best of our knowledge, there exist no systematic ways in the literature that fully address the burst scheduling problem for mobile TV networks with arbitrary and variable TV channel bit rates.

In [7] and [8], we have shown that the burst scheduling problem in mobile TV networks is NP-complete for TV channels with arbitrary bit rates. We also proposed a scheduling algorithm for a *special* case of the general problem: when the bit rates of the TV channels have power of 2 increments. In the current paper, we propose a near-optimal algorithm for the general burst scheduling problem that does not require any assumption on the channel bit rates: Both diverse and variable bit rates are supported. This algorithm provides great flexibility for the content providers to choose the appropriate encoding bit rates for different types of video content. This in turn will provide better utilization of the expensive wireless medium and thus more offered TV channels, as well as higher perceived video quality and thus wider adoption of the mobile TV service. In addition, the proposed algorithm enables the content provider to offer differentiated classes of services for different subscription rates. For example, better quality videos encoded at higher bit rates can be offered for higher premiums. This service differentiation is quite difficult (if at all possible) to offer with the current burst scheduling algorithms. The proposed algorithm achieves all of the above while making the energy consumption of mobile devices very close to the absolute possible minimum.

The contributions of this paper are as follows.

- We present a general formulation of the burst scheduling problem in mobile TV networks. This formulation is general because each TV channel can be coded at any arbitrary encoding rate and into either Constant-Bit-Rate (CBR) or Variable-Bit-Rate (VBR) streams. The objective of the formulation is to maximize the overall energy saving of all mobile devices, which is a critical issue as mobile devices are battery-powered and have stringent energy constraints.
- This general formulation is NP-complete and cannot be optimally solved within reasonable amount of time. We, therefore, propose an approximation algorithm to solve it. We prove the correctness of the proposed algorithm, and we show that it runs in time $O(pS \log pS)$, where $p$ is the scheduling frame length, and $S$ is the number of TV channels. Moreover, we prove that burst schedules produced by the proposed algorithm are very close to optimal schedules.
- We implement the proposed algorithm in a real mobile TV testbed, and we conduct extensive experiments using it. The experimental results show that the proposed algorithm: 1) is practical and produces correct burst schedules; 2) achieves near-optimal energy saving for mobile devices; and 3) is efficient and can run in real time.
- We also implement the proposed algorithm in a mobile TV simulator, which captures important aspects related to the burst scheduling problem, but abstracts away irrelevant details. We use this simulator to exercise the proposed algorithm with wider ranges of parameters that are difficult to set in the real testbed. The simulation results show that: 1) larger receiver buffer size allows higher energy saving, but incurs longer channel switching delay; 2) the proposed algorithm does scale to high network utilization with marginal performance degradation: energy saving reduces by less than 5% as the utilization approaches 100%; 3) a fairly small scheduling frame length, up to 60 s, results in high energy saving; and 4) the proposed algorithm is efficient even for extreme cases: it terminates in 100 ms under 100% network utilization.

The rest of this paper is organized as follows. Section II presents a brief background on mobile TV networks and reviews the related works in the literature. In Section III, we state the burst scheduling problem in mobile TV networks, and we present the mathematical formulation of the problem. We propose a new burst scheduling algorithm and analyze it in Section IV. In Section IV, we also explain how the proposed algorithm can be used to broadcast VBR streams in real mobile TV networks. We conduct extensive experiments and simulations and present the results in Section V. We conclude the paper in Section VI.

## II. Background and Related Work

### A. Mobile TV Networks

TV programs can be delivered to mobile devices using cellular networks or dedicated broadcast networks. In this paper, we focus on broadcast networks, which have the potential to serve to a large number of subscribers. There are several systems and standards for video broadcast networks, including T-DMB [9], ISDB-T [10], MediaFLO [4], and DVB-H [2], [11]. Among the above broadcast networks, only DVB-H and MediaFLO try to minimize the energy consumption of mobile devices by periodically turning their RF circuits off. MediaFLO [4] is a video broadcast system developed by Qualcomm. The details of the design are not public. In contrast, DVB-H [2], [11] is an open international standard. We use the open DVB-H standard in our discussion throughout the paper. Nonetheless, in our problem formulation and solution, we abstract away the specific details of the DVB-H standard. Therefore, our solutions are also applicable to other video broadcast networks.

We present an overview of the DVB-H standard. DVB-H is an extension to the Digital Video Broadcast-Terrestrial (DVB-T) standard [12] to support mobile devices. DVB-H standard defines protocols below the network layer and uses IP as the interface with the higher-layer protocols such as UDP and RTP. The IP Datacast standard [2] complements DVB-H by defining a set of higher-layer protocols for a complete end-to-end solution. DVB-H encapsulates IP packets using Multi-Protocol Encapsulation (MPE) sections to form MPEG-2 transport streams. Thus, data from a specific TV channel form a sequence of MPEs. MPEs are optionally FEC-protected before transmitted over the air medium. To save energy of mobile devices, MPEs belonging to a given TV channel are transmitted in bursts. Fig. 1 illustrates the main components of a DVB-H network. Our proposed burst

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

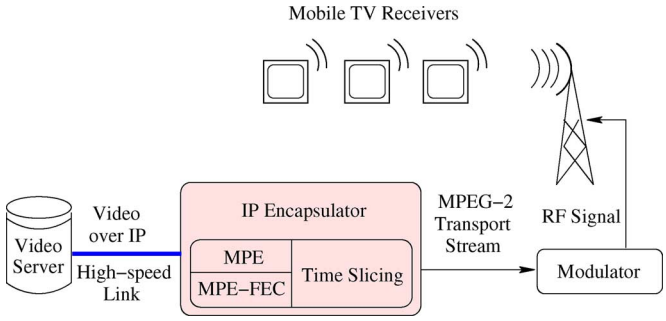


Fig. 1. The main components of mobile TV broadcast networks. Our work optimizes the time slicing component.

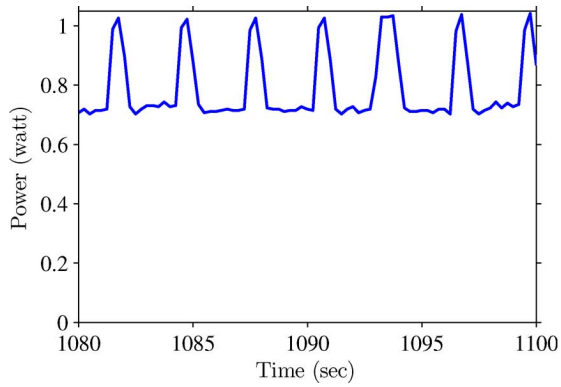


Fig. 2. Sample power consumption results captured on a Nokia N96 phone.

scheduling algorithm resides in the Time Slicing part of the IP Encapsulator, which runs on the base station as shown in Fig. 1.

We note that mobile devices have stringent battery capacity and heat dissipation requirements, thus they cannot accommodate mobile TV chips that consume too much energy. However, even the state-of-the-art prototype mobile TV chips, such as [13], consume 200 mW, while commercial chips consume as high as 500 mW in continuous mode [14]. These chips require significantly more power than the design guideline specified in the DVB-H standard [15], which states that the mobile TV chips with power consumption higher than 100 mW are not suitable for handheld devices. This gap emphasizes the importance of time slicing and the burst scheduling problem formulated and solved in the paper.

To understand the *actual* power consumption of the mobile TV chips in the market, we conduct an experiment using a real mobile TV testbed in our lab. More details on this testbed are given in Section V-A. We encode a 10-min TV news clip. We then use our mobile TV base station to broadcast this coded stream, and we restart it once its end is reached. We configure our base station to broadcast with a fixed interburst time period of 3 s. We use a Nokia N96 phone to watch this TV channel for 3.5 h, and we measure its energy consumption using a built-in energy monitor, called Juice. Juice is a lightweight program that runs in the background, which measures battery voltage, current, as well as power consumption and saves them into a log file. A recent study reports that Juice produces accurate measurements compared to external instruments [16]. We plot a sample power consumption curve in Fig. 2, which shows a small time window for readability; curve outside this time window is similar. This figure shows that there are periodical power consumption spikes of 0.3 W every 3 s, which equals the interburst time period. This means that the mobile TV chip used by Nokia N96 phones consumes 300 mW in continuous mode, which is about 30% of the total phone power consumption and is thus nontrivial.

### B. Related Work

A number of works have addressed energy saving in mobile TV networks. The authors of [15] and [17] estimate the effectiveness of the time slicing technique for given burst schedules. Both works indicate that time slicing enables mobile devices to turn off their RF circuits for a significant fraction of the time. These two works do not solve the burst scheduling problem; they only compute the achieved energy saving for a given *predetermined* burst schedule. In contrast, we formulate and solve the burst scheduling problem for arbitrary bit rates.

The authors of [18] propose an energy-saving strategy by not receiving some MPE-FEC sections once the received sections can successfully reconstruct the data. In this way, mobile devices can turn off their RF circuits earlier, which leads to additional energy saving compared to receiving all MPE-FEC sections. The authors of [19] consider mobile devices with an auxiliary short-range wireless interface and construct a cooperative network among several devices over this short-range wireless network. Mobile devices share received IP packets over this short-range network, so that each mobile device only receives a small fraction of IP packets directly from the DVB-H network. This allows these mobile devices to reduce the frequency of opening their RF circuits. Assuming sending/receiving IP packets through the short range network is more energy-efficient than receiving DVB-H sections, this cooperative strategy can save energy consumption. The proposals in [18] and [19] are orthogonal and complementary to our work as they reside in the mobile devices themselves and try to achieve additional energy saving on top of that achieved by time slicing. In contrast, our algorithm is implemented in the base station broadcasting TV channels to mobile devices.

We note that receivers in mobile TV *broadcast* networks have separate RF circuits and antennas for processing TV signals, other than the circuits for receiving and making phone calls. Our work focuses only on optimizing the energy saving for TV signal receivers. In addition, because of the one-way nature of the broadcast networks, feedback channels from numerous receivers to the base station are not practical. Thus, many of the energy-saving techniques designed for video streaming to the general wireless devices are not readily applicable to mobile TV networks. For example, the throttling technique proposed in [20], which enables a wireless receiver to indirectly control the sending pattern of an Internet streaming server, requires a feedback channel from the receiver to the server, which may not be possible in mobile TV networks.

Finally, our previous work proposed an optimal burst scheduling algorithm for a *simplified* version of the burst scheduling problem where TV channels are classified into a few classes, and each class has a different bit rate [7], [8]. The bit rate of class $c$, $r_c$, can take any value in the form of $r_c = 2^i \times r_1$, where

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

TABLE I
LIST OF SYMBOLS USED

| Sym | Definition | Sym | Definition |
|---|---|---|---|
| $T_o$ | overhead duration | $n_s$ | no. bursts for $s$ |
| $S$ | number of channels | $f_s^k$ | time of burst $k$ for $s$ |
| $R$ | burst bit rate | $b_s^k$ | burst size |
| $Q$ | receiver buffer size | $c_s^k$ | buffer level |
| $r_s$ | channel bit rate for $s$ | $u_s$ | init. buffer level |
| $p$ | frame length | $\gamma$ | energy saving |
| $\mathbf{L}$ | burst schedule | $d$ | ch. switching delay |
| $\mathbf{w}_s$ | set of subframs for $s$ | $w_s^k$ | subframe $k$ for $s$ |
| $x_s^k$ | start time of $w_s^k$ | $z_s^k$ | end time of $w_s^k$ |
| $y_s^k$ | total burst time of $w_s^k$ | $e_s^k$ | completion time of $w_s^k$ |

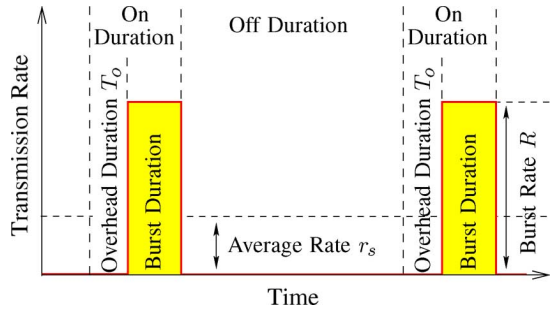


Fig. 3. Time slicing in mobile TV networks to save energy.

$i \in \{0, 1, 2, 3, \ldots\}$, and $r_1$ is the bit rate of the lowest class. In this paper, we study the general scheduling problem where each TV channel can take any arbitrary bit rate. This general problem enables finer-grained optimization by providing higher bit rate flexibility. Preliminary results of this work were published in [21].

## III. BURST SCHEDULING PROBLEM

In this section, we state the burst scheduling problem, and we mathematically formulate it. The notations used in the paper are listed in Table I.

### A. Problem Statement

We consider mobile TV networks in which a base station broadcasts $S$ digital TV channels to mobile devices over a shared air medium with bandwidth $R$ kbps. Examples of such networks include DVB-H [2], [3] and MediaFLO [4]. Each TV channel $s$, $1 \leq s \leq S$, has a bit rate $r_s$ kbps. We consider a very general problem where each TV channel can be encoded in arbitrary bit rate suitable to the video content carried by this channel. To save the energy consumption of mobile devices, the base station broadcasts each TV channel in *bursts* at bit rate $R$ kbps. Thus, after receiving and buffering a burst of data, mobile devices can switch off their RF circuits till the next burst. The next burst time is computed by the base station and included in the header fields of every burst. This is referred to as time slicing, especially in the terminology of the DVB-H standard [2], [3]. Fig. 3 depicts time slicing in mobile TV networks. Notice that the RF circuits are turned on slightly before the burst time because it takes some time to wake up and synchronize the circuitry before it can start receiving data. This time is called the overhead duration and is denoted by $T_o$. $T_o$ is in the range of 50–250 ms with current technology [2], [15].

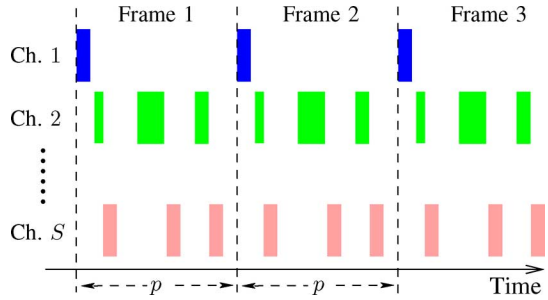


Fig. 4. The burst scheduling problem in mobile TV networks.

Let $\gamma_s$ be the energy saving of the mobile devices receiving TV channel $s$. $\gamma_s$ is calculated as the ratio of the time the RF circuits are in off mode to the total time. The $\gamma_s$ values indicate the energy saving due to time slicing, and it has been used by previous works in the literature [17], [18] and in the standardization documents [15]. We define the average system-wide energy saving over all TV channels as $\gamma = (\sum_{s=1}^{S} \gamma_s)/S$. The energy saving as well as the time slicing itself are performed on a recurring time window called a frame. We let $p$ denote the frame length, which is a system parameter in mobile TV broadcast networks. Fig. 4 shows an example of three recurring frames. In general, longer frame lengths provide more chances to shuffle bursts around for better energy saving. However, a longer $p$ may increase the channel switching delay and computation complexity of the burst scheduling algorithm. We will empirically study these tradeoffs in Section V.

Now, the burst scheduling problem can be stated as follows.

*Problem 1 (Burst Scheduling in Mobile TV Systems):* Given $S$ TV channels of different and variable bit rates to be simultaneously broadcast to mobile devices. Each TV channel is broadcast as bursts of data to save the energy consumption of mobile devices. Our problem is to find the optimal transmission schedule for bursts of all TV channels to maximize the system-wide energy saving $\gamma$. The transmission schedule specifies the number of bursts for each TV channel in a frame $p$ as well as the start and end times for each burst. The schedule cannot have burst collisions, which happen when two or more bursts have nonempty intersection in time. In addition, the schedule must ensure that there are no receiver buffer violations for any channel. A buffer violation occurs when the receiver of a TV channel either: 1) has no data in the buffer to playout (buffer underflow); or 2) has no space to store data during a burst transmission (buffer overflow). □

### B. Problem Formulation

We show a simple example of the burst scheduling problem in Fig. 4. This figure depicts that, in the considered problem, the bursts have various sizes, are disjoint in time, and are repeated in all recurring frames. Furthermore, a TV channel can have multiple bursts in each frame to ensure that there are no buffer violations. To illustrate the receiver buffer dynamics for a valid schedule, we demonstrate in Fig. 5 the receiver buffer level of a TV channel with two bursts in each frame. We make two

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

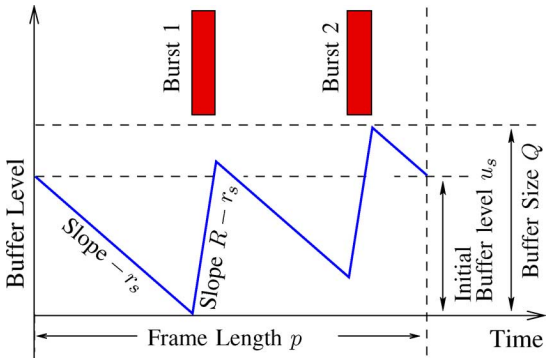


Fig. 5. Dynamics of the receiver buffer.

observations about this figure. First, during a burst, the buffer level increases with a rate (slope of the line) of $R-r_s$, which is much larger than the consumption rate of $-r_s$ when there is no burst. Second, the frame starts with an initial buffer level (denoted by $u_s$) and ends at the same buffer level. Clearly, this is a requirement for any valid burst scheduling solution, otherwise the receiver buffer may have over/underflow instances.

Let $n_s$ be the number of bursts of TV channel $s$ in each frame. We denote the start time and burst size of burst $k$ of channel $s$ as $f_s^k$ s and $b_s^k$ kb, respectively, where $s = 1, 2, \ldots, S$ and $k = 1, 2, \ldots, n_s$. Since mobile devices open their RF circuits $T_o$ ms before $f_s^k$ and it takes $b_s^k/R$ to transfer $b_s^k$ kb data, the RF circuits are on for burst $k$ of channel $s$ during time period $[f_s^k - T_o, f_s^k + b_s^k/R)$. In addition, any burst $b_s^k$ must be smaller than the receiver buffer size $Q$, i.e., $0 < b_s^k \leq Q$. We define the buffer level at the beginning of burst $k$ of TV channel $s$ as $c_s^k$ kb. As illustrated in Fig. 5, $c_s^k$ can be computed as

$$c_s^k = u_s + \sum_{i=1}^{k-1} b_s^i - f_s^k r_s$$

where the second term accounts for the received data and the third term accounts for the consumed data. The output of the burst scheduling algorithm is a schedule $\mathbf{L} = \{\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_S\}$, where $\mathbf{L}_s$ is the schedule for TV channel $s$. In addition, $\mathbf{L}_s = \langle n_s, u_s, \mathbf{f}_s, \mathbf{b}_s \rangle$, where $n_s$ is the number of bursts, $u_s$ is the initial buffer level, $\mathbf{f}_s = \{f_s^1, f_s^2, \ldots, f_s^{n_s}\}$ indicates the burst start times, and $\mathbf{b}_s = \{b_s^1, b_s^2, \ldots, b_s^{n_s}\}$ represents the burst sizes.

The burst scheduling problem can then be formulated as

$$\max_{\mathbf{L}} \quad \gamma = \sum_{s=1}^{S} \left(1 - \sum_{k=1}^{n_s} \left(T_o + b_s^k/R\right)/p\right)/S \tag{1a}$$

$$\text{s.t.} \quad \left[f_s^k, f_s^k + \frac{b_s^k}{R}\right) \bigcap \left[f_{\bar{s}}^{\bar{k}}, f_{\bar{s}}^{\bar{k}} + \frac{b_{\bar{s}}^{\bar{k}}}{R}\right) = \varnothing \tag{1b}$$

$$c_s^k \geq 0 \tag{1c}$$

$$c_s^k + b_s^k - \frac{b_s^k}{R} r_s \leq Q \tag{1d}$$

$$0 \leq u_s \leq Q \tag{1e}$$

$$\sum_{i=1}^{n_s} b_s^i = p r_s \tag{1f}$$

$$\forall 1 \leq s \neq \bar{s} \leq S, \quad 1 \leq k \leq n_s, \quad 1 \leq \bar{k} \leq n_{\bar{s}}.$$

The goal is to compute the schedule $\mathbf{L}$ to maximize the objective function in (1a), i.e., the system-wide energy saving $\gamma$. The constraints (1b)–(1f) guarantee that the resulting burst schedule is feasible as defined in Problem 1. In particular, (1b) ensures that there are no burst intersections among all $S$ channels. (1c) validates the buffer level for channel $s$ at the start time of every burst to prevent buffer underflow instances. We note that $c_s^k$ is a function of $f_s^k$, $b_s^k$, and $u_s$ as defined above. (1d) validates the buffer level for channel $s$ at the end time of every burst to prevent buffer overflow instances. It is sufficient to check the buffer level only at the start and end times because the buffer level only increases during the bursts, as illustrated in Fig. 5. The buffer under- and overflow instances at frame boundaries are prevented by (1e). (1f) says that the number of received and consumed bits for channel $s$ are equivalent in every frame, which in turn ensures that the buffer level at the end of every frame is equal to the initial buffer level $u_s$.

### C. Hardness

The considered burst scheduling problem is fairly general and quite difficult to solve. In fact, we have proved that the burst scheduling problem is NP-complete in our previous work [7], [8]. Moreover, although the considered problem might look somewhat similar to preemptive machine scheduling problems at a first glance, there is a fundamental difference between our burst scheduling problem and various machine scheduling problems. Most machine scheduling problems consider *costless* preemption model [22], while our burst scheduling problem adopts *costly* preemption model as each preemption in our problem leads to energy consumption overhead for $T_o$ ms period, which is not negligible compared to the burst size. The costly preemption model has only been considered in a few works [23]–[26]. The authors of [24] and [25] partially cope with preemption costs by adding constraints to limit the number of preemptions. The authors of [23] solve the problem of minimizing the weighted sum of the total task flow time and the preemption penalty, where the weight is *ad hoc*. The author of [26] considers the problem of minimizing weighted completion time and task makespan under a given preemption cost. Unlike these works that partially cope with preemption costs, our burst scheduling problem considers preemption cost in the objective function and does not allow any overdue bursts. Hence, these algorithms, as well as others developed in the literature with costless preemption model [22], are not applicable to the burst scheduling problem. Thus, we develop an approximation algorithm to solve the burst scheduling problem in Section IV.

## IV. Near-Optimal Algorithm

We first observe that the formulation in (1) has two *tightly coupled* constraints: no burst intersection [(1b)] and no receiver buffer violation [(1c)–(1f)]. These two constraints prevent us from employing many common techniques for solving machine scheduling problems. Therefore, in Section IV-A, we propose to *decouple* these two constraints by transforming the *original* formulation in (1) into another formulation, which only has *one* constraint and is easier to solve. We call the new formulation

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

*transformed* formulation. In Section IV-B, we present an efficient algorithm to solve the transformed formulation with near-optimality. We also convert the near-optimal schedule for the transformed formulation back to the original formulation in the same subsection. In Section IV-C, we analytically study the proposed algorithm, and we show its correctness, complexity, and approximation factor.

### A. Transform

The goal of the transform is to construct a formulation with a single constraint of no burst collisions. The transformed formulation is then similar to those of the machine scheduling problems and can be efficiently solved with near-optimality. To achieve this goal, we propose to split the receiver buffer into two equal-sized buffers, called $B$ and $B'$, and we divide the frame into multiple subframes such that the number of bits that is received for smooth playout in each subframe never exceeds the size of $B$ or $(B')$. In every subframe, a mobile device stores the received data in one of its buffers, say $B$, for later usage and decodes the previously received data from another buffer, say $B'$. The mobile device swaps its two buffers upon a new subframe is reached: It stores the received data in the empty buffer $B'$ and decodes the data from the filled buffer $B$. For smooth playouts, we need to make sure that the number of received bits in each subframe is the same as the number of consumed bits in the subsequent subframe, and we present a systematic way of doing this in the following.

We let $\mathbf{w}_s$ be a set of subframes of TV channel $s$, where $s = 1, 2, \ldots, S$. More specifically, we define $\mathbf{w}_s = \{w_s^k \mid k = 1, 2, \ldots, \lceil 2pr_s/Q \rceil\}$ and $w_s^k = \langle x_s^k, y_s^k, z_s^k \rangle$, where

$$x_s^k = (k-1)Q/(2r_s) \tag{2a}$$

$$z_s^k = \begin{cases} kQ/(2r_s), & 1 \le k \le \lfloor 2pr_s/Q \rfloor \\ p, & k = \lceil 2pr_s/Q \rceil \text{ if } 2pr_s/Q \notin \mathbb{Z} \end{cases} \tag{2b}$$

$$y_s^k = \begin{cases} Q/(2R), & 1 \le k \le \lfloor 2pr_s/Q \rfloor \\ \frac{Q}{2R} \times \frac{p \bmod \frac{Q}{2r_s}}{\frac{Q}{2r_s}}, & k = \left\lceil \frac{2pr_s}{Q} \right\rceil \text{ if } \frac{2pr_s}{Q} \notin \mathbb{Z}. \end{cases} \tag{2c}$$

In this definition, $w_s^k$ is subframe $k$ of TV channel $s$ and is written as a 3-tuple $\langle x_s^k, y_s^k, z_s^k \rangle$, where $x_s^k$ and $z_s^k$ are the start and end times of this subframe, and $y_s^k$ is the total burst time that should be assigned to the TV channel between $x_s^k$ and $z_s^k$ for smooth playouts. With the above notations, we write the transformed formulation as

$$\min_{\mathbf{L}} \quad \sum_{s=1}^{S} n_s \tag{3a}$$

$$\text{s.t.} \quad \left[ f_s^{\hat{k}}, f_s^{\hat{k}} + \frac{b_s^{\hat{k}}}{R} \right) \bigcap \left[ f_{\bar{s}}^{\bar{k}}, f_{\bar{s}}^{\bar{k}} + \frac{b_{\bar{s}}^{\bar{k}}}{R} \right) = \varnothing \tag{3b}$$

$$\sum_{x_s^k \le f_s^i \le z_s^k} b_s^i = R y_s^k \tag{3c}$$

$$\forall 1 \le s \ne \bar{s} \le S, 1 \le k \le \lceil 2pr_s/Q \rceil, \quad 1 \le \hat{k} \le n_s, 1 \le \bar{k} \le n_{\bar{s}}.$$

In this formulation, the objective function in (3a) minimizes the number of total bursts in the frame. This is because fewer bursts lead to less overhead due to waking up RF circuits and thus result in higher energy saving. The constraint in (3b) prevents any intersected bursts, and the constraint in (3c) ensures that the total burst length scheduled for TV channel $s$ between times $x_s^k$ and $z_s^k$ *equals* to $y_s^k$. We notice that the rationale behind the definition of $\mathbf{w}_s$ is to make sure that every subframe $w_s^k$ gets scheduled bursts that are sufficient to transmit $Q/2$ kb video data [indicated by (2c)], which equals to the amount of data required for smooth playouts in the subsequent subframe [indicated by (2a) and (2b)]. In addition, while each subframe $w_s^k$ ($s = 1, 2, \ldots, S$ and $k = 1, 2, \ldots, \lceil 2pr_s/Q \rceil$) requires total burst time $y_s^k$, it need *not* be from a *single* burst.

### B. Efficient Algorithm

We present an efficient algorithm to solve the transformed formulation in (3) in the sequel, and we call it Double Buffering Scheduling (DBS) algorithm. We define *decision points* as the time instances at which either a new subframe starts, i.e., at time $x_s^k$ for any $s$ and $k$, or bursts scheduled to a subframe have met the required burst length, i.e., satisfying the constraint in (3c). At each decision point $t$, our scheduling algorithm schedules a burst for the subframe with the smallest end time $z_s^k$ among all outstanding subframes with start time earlier than the current time, i.e., $x_s^k \le t$. We say a subframe is outstanding if and only if it requires more bursts, i.e., the current schedule has not satisfied its constraint in (3c) yet. Moreover, we let $e_s^k$ be the completion time of subframe $w_s^k$, where $e_s^k$ represents the time at which its constraint in (3c) is met. Our scheduling algorithm terminates whenever there is no outstanding subframe nor a subframe that has a start time in the future. In Lemma 1, we prove that our algorithm gives a burst schedule that minimizes the subframe lateness, which is defined as $e_s^k - z_s^k$. The achieved lateness allows us to determine whether the resulting burst schedule is feasible or not. More precisely, we check whether $e_s^k - z_s^k$ is nonpositive for all $s$ and $k$. If this is true, we know that the resulting burst scheduling is feasible. Otherwise, our algorithm prompts the network operator to reduce the number of TV channels because the required bandwidth exceeds the network capacity.

Fig. 6 gives a high-level pseudo code of the proposed DBS burst scheduling algorithm. In lines 1–4, the algorithm constructs the transformed formulation by defining the subframes for all TV channels. In lines 5–9, it traverses through all decision points in the transformed formulation and builds a burst schedule. Last, it checks whether the resulting schedule is feasible in lines 10–13.

### C. Analysis

We first show, in the next lemma, that the DBS algorithm can solve the transformed formulation.

*Lemma 1:* The DBS algorithm finds a burst schedule for the transformed formulation in (3) if and only if one exists.

*Proof:* First, we prove that if the DBS algorithm returns a schedule $\mathbf{L}$, then $\mathbf{L}$ is a feasible solution for the formulation in (3). In Fig. 6, the for-loop between lines 7–9 iterates through all decision points, which are defined as the time instances at which a subframe starts or a subframe reaches its required burst length. While at each decision point, line 8 schedules the outstanding subframe with the smallest frame end time. Therefore,

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

**Double Buffering Scheduling (DBS)**

1. // construct the transformed formulation
2. **for** $s = 1$ to $S$
3. **for** $k = 1$ to $\lceil 2pr_s/Q \rceil$
4. determine $x_s^k$, $y_s^k$, and $z_s^k$ using Eqs. (2a)—(2c)
5. // schedule bursts
6. **let** $\mathbf{L} = \varnothing$
7. **foreach** decision point {
8. add a burst from times $t_c$ to $t_n$ to channel $s$, where $\mathbf{w}_s^k$
8. has the smallest $z_s^k$ among outstanding subframes, $t_c$
8. is current time, and $t_n$ is time of the next decision point
9. }
10. // feasibility check
11. **if** $\max\{e_s^k - z_s^k\} \leq 0$ // complete on time
12. **return** $\mathbf{L}$
13. **return** no feasible schedule

Fig. 6. A near-optimal burst scheduling algorithm.

resulting schedules of the DBS algorithm have no burst collisions and satisfy (3b). We notice that this for-loop actually composes a burst schedule $\mathbf{L}$ that minimizes the maximal subframe lateness $e_s^k - z_s^k$ among all subframes of all TV channels. This can be proved by transforming any optimal burst schedule to $\mathbf{L}$ with a finite number of burst swaps without compromising the schedule feasibility. This burst swapping technique is similar to the one used in [22, Theorem 4.4], which solves a single machine preemptive scheduling problem for minimizing task lateness. In lines 11–13, the algorithm checks whether $\mathbf{L}$ satisfies (3c) by comparing the maximal subframe lateness against zero. Since $\mathbf{L}$ returned in line 13 satisfies (3), the proof follows.

Second, we show that if the algorithm does not find a feasible burst schedule, there exists no solution for the formulation in (3). We prove this by contradiction. Assume that there is a feasible schedule $\hat{\mathbf{L}}$ and our algorithm in line 13 claims the resulting $\mathbf{L}$ is not feasible. Note that line 13 says that $\mathbf{L}$ is not feasible only if the corresponding maximal subframe lateness is positive. However, the feasible schedule $\hat{\mathbf{L}}$ must have a nonpositive maximal subframe lateness, and the maximal subframe lateness of $\hat{\mathbf{L}}$ is smaller than that of $\mathbf{L}$. However, as discussed above, the for-loop between lines 7–9 minimizes the maximal subframe lateness, which leads to a contradiction because the algorithm should have returned $\hat{\mathbf{L}}$ instead. ■

We then show in the next lemma that the proposed transform does not affect the existence of feasible burst schedules.

*Lemma 2:* There exists a feasible schedule for the transformed formulation in (3) if and only if there exists a feasible schedule for the original formulation in (1).

*Proof:* We claim that for an arbitrary buffer size $\bar{Q}$, there exists a feasible burst schedule for the original burst scheduling formulation in (1) if and only if $\sum_{s=1}^{S} r_s \leq R$. We show this by constructing a feasible schedule as follows. Without loss of generality, we assume that $r_1 \leq r_2 \leq \cdots \leq r_S$. We allocate a burst to each TV channel every $\bar{Q}/r_S$ s, the burst size for TV channel $s$ is $(\bar{Q}/r_S) \times r_s$, and bursts for different TV channels are placed in a round-robin fashion. We note that the resulting schedule leads to no buffer violations for any TV channel $s$. More importantly, the total burst length in each round $(\sum_{s=1}^{S}(\bar{Q}/r_S) \times (r_s/R))$ is no longer than the round duration $(\bar{Q}/r_S)$ if and only if $\sum_{s=1}^{S} r_s \leq R$. This shows that the existence of a feasible burst schedule for the original formulation in (1) is independent from the receiver buffer size $Q$. Hence, the proposed transform does not affect the existence of feasible schedules. ■

With these two lemmas, we can now show that the DBS algorithm always finds a feasible burst schedule for the original burst scheduling formulation in (1).

*Theorem 1 (Correctness):* The DBS algorithm produces a feasible schedule for the original formulation in (1).

*Proof:* Lemmas 1 and 2 reduce the proof to showing that any feasible schedule for the formulation in (3) is also feasible for the formulation in (1). In lines 2–4, we divide each frame into $\lceil 2pr_s/Q \rceil$ disjoint subframes, where each subframe has a required burst length $Q/2$ kb. Therefore, following the definition of $y_s^k$ in (2c), any feasible burst schedule $\mathbf{L}$ for the transformed formulation in (3) transmits $Q/2$ kb data in every subframe. Furthermore, the definitions of $x_s^k$ and $z_s^k$ in (2a) and (2b) indicate that the duration of each subframe is $Q/(2r_s)$ and the amount of data to support smooth playout in a subframe is $Q/2$. Finally, letting $u_s = Q/2$ $(s = 1, 2, \ldots, S)$ gives a burst schedule for the original formulation in (1), where mobile devices always consume the data received in the immediate, preceding subframe. Therefore, this schedule satisfies the constraints in (1c)–(1f), which yields the theorem. ■

This theorem completes our proof of correctness. Next, we derive the approximation factor of the DBS algorithm and its time complexity.

*Theorem 2 (Approximation Factor and Time Complexity):* The DBS algorithm produces near-optimal burst schedules with the approximation factor

$$\frac{\gamma^*}{\gamma} \leq \frac{1 - T_o R/SQ - 1/S}{1 - 2T_o R/SQ - 1/S} \tag{4}$$

where $\gamma^*$ and $\gamma$ are the average energy saving achieved by the optimal scheduling algorithm and the DBS algorithm, respectively. Moreover, the DBS algorithm runs in time $O(pS \log(pS))$.

*Proof:* To compute the approximation factor, we first determine the number of bursts in the optimal schedule $\mathbf{L}^*$ and in the schedule $\mathbf{L}$ that is produced by the DBS algorithm. Consider any transformed formulation constructed by the for-loop in lines 2–4 with a set of subframes $\mathbf{w}$. The number of subframes is given as $|\mathbf{w}| = \sum_{s=1}^{S} \lceil 2pr_s/Q \rceil$. Notice that fewer preemptions in general lead to higher energy saving. However, to prevent buffer overflow instances, any optimal burst schedule $\mathbf{L}^*$ must have at least $\lceil pr_s/Q \rceil$ bursts for each TV channel $s$, i.e., $n_s^* \geq \lceil pr_s/Q \rceil$. Thus, we have $|\mathbf{w}| \leq 2|\mathbf{L}^*|$. We then consider the number of bursts produced by the DBS algorithm. We note that the total number of bursts is bounded by the number of decision points, which are defined as the time instances at which either a subframe starts or completes. Observe that, except for the boundary cases, a new subframe is only *created* when the previous subframe *completes*. This means that the total number of decision points is $|\mathbf{w}| + S \cong |\mathbf{w}|$. Since $|\mathbf{w}| \leq 2|\mathbf{L}^*|$, the number of bursts in $\mathbf{L}$ is at most two times the number of bursts in the optimal schedule $\mathbf{L}^*$. Thus, we have $\sum_{s=1}^{S} n_s \leq 2 \sum_{s=1}^{S} n_s^*$.

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

Now, we derive the approximation factor $\gamma^*/\gamma$ as follows. Following the definition of the average energy saving $\gamma$, we write the optimal energy saving as

$$\gamma^* = 1 - \frac{T_o}{pS}\sum_{s=1}^{S} n_s^* - \frac{1}{RS}\sum_{s=1}^{S} r_s \approx 1 - \frac{T_o}{S}\frac{\sum_{s=1}^{S} r_s}{Q} - \frac{\sum_{s=1}^{S} r_s}{RS}. \quad (5)$$

Similarly, we write the energy saving achieved by our algorithm as

$$\gamma = 1 - \frac{T_o}{pS}\sum_{s=1}^{S} n_s - \frac{1}{RS}\sum_{s=1}^{S} r_s \geq 1 - 2\frac{T_o}{pS}\sum_{s=1}^{S} n_s^* - \frac{1}{RS}\sum_{s=1}^{S} r_s \approx 1 - 2\frac{T_o}{S}\frac{\sum_{s=1}^{S} r_s}{Q} - \frac{\sum_{s=1}^{S} r_s}{RS}. \quad (6)$$

While combining these two inequalities gives the approximation factor $\gamma^*/\gamma$, the resulting equation is too complex to reveal useful insights. To simplify it, we assume the bandwidth of the mobile TV network is saturated: $R \approx \sum_{s=1}^{S} r_s$. This is a reasonable assumption because the wireless spectrum is expensive and service providers would broadcast as many TV channels as possible. With this practical assumption, we get (4).

Last, we analyze the time complexity of the DBS algorithm. Observe that constructing the set $\mathbf{w}$ takes $O(|\mathbf{w}|)$, sorting subframes on start times takes $O(|\mathbf{w}|\log|\mathbf{w}|)$, and storing outstanding subframes in the priority queue and composing $\mathbf{L}$ take $O(|\mathbf{w}|\log|\mathbf{w}|)$. Moreover $|\mathbf{w}| = \sum_{s=1}^{S}(2pr_s/Q)$, and $r_s/Q$ can be considered as a small constant for practical encoding bit rates (few hundreds of kbps) and buffer sizes (few Mb). Thus, the time complexity of the DBS algorithm is $O(|\mathbf{w}|\log|\mathbf{w}|) = O(pS\log(pS))$. ■

To shed some light on the approximation factor derived in the above theorem, we numerically analyze it using a range of practical values. We consider mobile TV networks with various wireless medium capacities between 4.354 and 19.595 Mbps. These values cover possible bit rates of a 7-MHz frequency band with different modulation and coding schemes [15]. We let the overhead duration $T_o = 0.1$ s. We first fix receiver buffer size at $Q = 2$ Mb and vary number of TV channels between 10 to 40. We compute the approximation factor $\gamma^*/\gamma$ and plot it in Fig. 7 for different capacities of the wireless medium. The figure shows that our DBS algorithm produces very close results to the optimal ones. For example, using our algorithm to broadcast 10 to 40 TV channels over a 7.620 Mbps medium, the average energy saving achieved by mobile devices is about 5% less than the absolute maximum energy saving that can be achieved using any algorithm to solve this NP-complete problem. Also, as detailed in the evaluation section, our algorithm obtains these near-optimal results in the order of tens of milliseconds on a commodity PC. Notice also that as the number of TV channels increases, the approximation factor of our algorithm actually improves and approaches 1. Next, we analyze the approximation factor as the receiver buffer $Q$ varies from 1 to 16 Mb, while the number of TV channels is fixed at 30. The results (not shown here due to space limitations) confirm that the approximation factor is typically close to 1, and it becomes even closer to 1 as the receiver buffer size increases, which is an expected trend in the future. These numerical results imply that our algorithm will yield almost optimal results in most deployments of mobile TV networks.

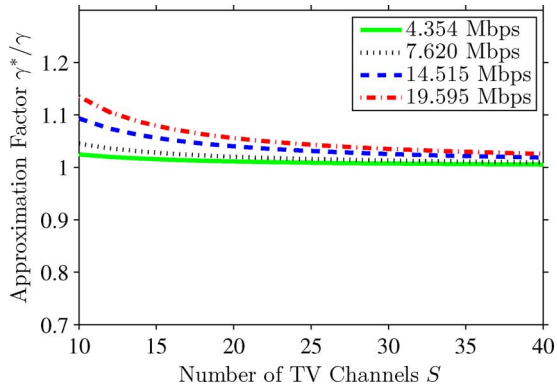


Fig. 7. The approximation factor of the DBS algorithm.

### D. Broadcasting Rate-Regulated VBR Video Streams

So far in this paper, we considered multiple TV channels encoded in different bit rates, and the bit rate of each channel is constant. In this section, we explain how our algorithm handles the variability in the bit rates of TV channels.

TV programs can be encoded into either CBR or VBR streams, where CBR coders dynamically adjust the coding parameters (such as the quantization step) to produce streams with constant bit rates, while VBR coders may fix the coding parameters to produce streams with constant video quality. Strict CBR streams, in which every picture is encoded with a fixed number of bits, suffer coding inefficiency and high quality fluctuation [27]. Therefore, modern video coding standards such as H.264/AVC [28] support CBR coding in a relaxed sense, where streaming servers employ a rate-regulation buffer to *regulate* VBR coded streams into CBR streams at streaming time. We describe the rate-regulation process in the following.

The rate-regulation operation can be described by the Hypothetical Reference Decoder (HRD) model of H.264/AVC [29], which guides streaming systems to properly set up buffer to smooth out the streaming traffic without incurring any buffer under/overflow instances for smooth playouts. Similar to hypothetical reference decoders defined in previous video coding standards, the H.264/AVC HRD model is based on the leaky bucket model. Fig. 8 illustrates the buffer dynamics at both the sender (on the left) and the receiver (on the right), which are connected by a CBR communication channel with a constant delay. At the sender side, each coded picture is instantaneously inserted into the sender's buffer, where the coded pictures can be generated by an online video coder or read from a stream file that was generated offline. Since coded pictures can have different sizes, the accumulated data amount of the video stream is a staircase, where the height of steps are varying. The buffered data is *drained* and sent to the receiver at a *constant* bit rate of $r$, where $r$ is the slope of the straight line beneath the staircase,

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

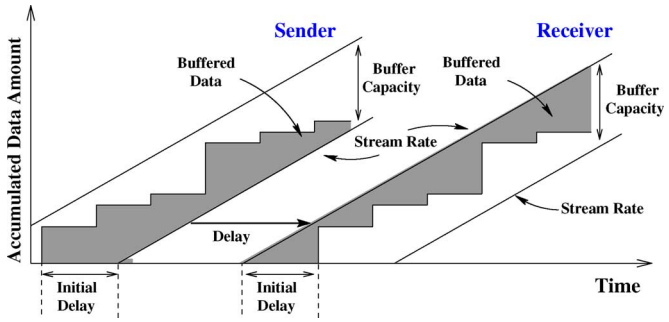


Fig. 8. Dynamics of the rate-regulation buffers of sender and receiver.

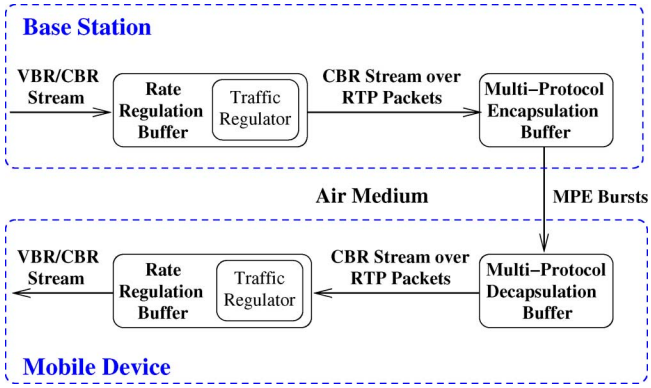


Fig. 9. Rate regulator and other buffers in mobile TV networks.

and the area below the staircase (shaded in the figure) represents the remaining buffered data in the sender's buffer.

After a constant transmission delay, the data arrives at the receiver also at bit rate $r$ and is stored in the receiver's buffer. Since coded pictures have various sizes, the video decoder at the receiver waits for enough number of bits and instantaneously removes each coded picture from the receiver's buffer for decoding, which is represented as another staircase in this figure. The area above this staircase (shaded in the figure) represents the remaining buffered data in the receiver's buffer. We note that the upper-left line at the sender side indicates the sender's buffer limit and the lower-right line at the receiver side indicates the receiver's buffer limit.

Note that no buffer under/overflow is possible if the staircase (at either sender or receiver) stays within the tube of the two straight lines, and the tube can be uniquely specified by its streaming rate, buffer capacity, and initial delay [29]. Therefore, streaming systems, including mobile TV networks, can employ the HRD model and use rate-regulation buffer to regulate the streaming rates of video streams, although the pictures were not coded in uniform size. We call VBR streams that can be rate-regulated into CBR streams *rate-regulated* VBR streams, which are also known as *shaped* VBR streams [27].

Rate-regulated VBR streams can be broadcast in mobile TV networks, as illustrated in Fig. 9. At the base station, from the left to right, the coded pictures are inserted into a Rate-Regulation Buffer that adopts a traffic regulator to produce a CBR stream at a constant streaming rate. This CBR stream is sent over RTP packets to the Multi-Protocol Encapsulation Buffer, where they form MPE bursts, which are larger than the RTP packets. MPE bursts are then broadcast over the air medium and stored in Multi-Protocol Decapsulation Buffer before being decapsulated back to a CBR stream at the mobile device. Finally, the CBR stream is converted back to the original video stream by the traffic regulator in the Rate-Regulation Buffer on the mobile device. The burst scheduling algorithm proposed in the paper manages the Multi-Protocol Encapsulation Buffer, which resides *after* the Traffic Regulator at the base station. We mention that a buffer setup similar to Fig. 9 is mentioned in DVB-H standard documents [30, Sec. 5.1.3] and is being used in real systems.

Next, we discuss the issues of broadcasting rate-regulated VBR streams. To get rate-regulated VBR streams, users need to specify the streaming rates, buffer capacity, and initial delays at encoding time, which instruct video coders to constrain the size of each coded picture in order to avoid over/underflowing the rate-regulation buffer. Broadcasting the resulting rate-regulated VBR streams may lead to a couple of drawbacks. First, they are not *unconstrained* VBR streams, as the coded picture size may be limited by the leaky bucket tubes illustrated in Fig. 8. Nonetheless, the coding efficiency of rate-regulated VBR streams is still better than that of strict CBR streams [27]. Second, the rate-regulation process requires a rate-regulation buffer and incurs longer initial delay. However, both the memory requirement and the initial delay are *user-specified* and can thus be controlled in reasonable ranges. Therefore, using mobile TV networks illustrated in Fig. 9 to broadcast TV programs coded as rate-regulated VBR streams is feasible and leads to better quality of service than broadcasting CBR streams. Nevertheless, a better solution is to develop a burst scheduling algorithm that directly handles *unconstrained* VBR streams for high energy saving, which remains one of our future works. Broadcasting unconstrained VBR streams is one of the most critical and challenging open problems in mobile TV broadcast networks [31], [32].

## V. Evaluation

In this section, we conduct extensive experiments using a real mobile TV testbed as well as simulations to validate the correctness, efficiency, and near-optimality of the proposed burst scheduling algorithm. We start by presenting the results from our testbed. Then, we use simulations to analyze wider ranges of several system parameters and their impact on the performance.

### A. Setup of the Mobile TV Testbed

We have set up a complete testbed for mobile TV networks. The testbed consists of two parts, a base station and receivers, which are described in the following. We use a commodity Linux PC as the base station and install a PCI modulator [33] in it. The modulator implements the physical layer of the DVB-H standard and is connected to an indoor antenna via a low-power amplifier. In order to drive the modulator to transmit DVB-H compliant signals, we run video server, IP encapsulator, and modulator software in the base station. Some of the software components are developed by us, and others are leveraged from open-source projects. In the IP encapsulator, we have designed the time slicing module to be well structured with clear interfaces in order to facilitate various burst scheduling algorithms.

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

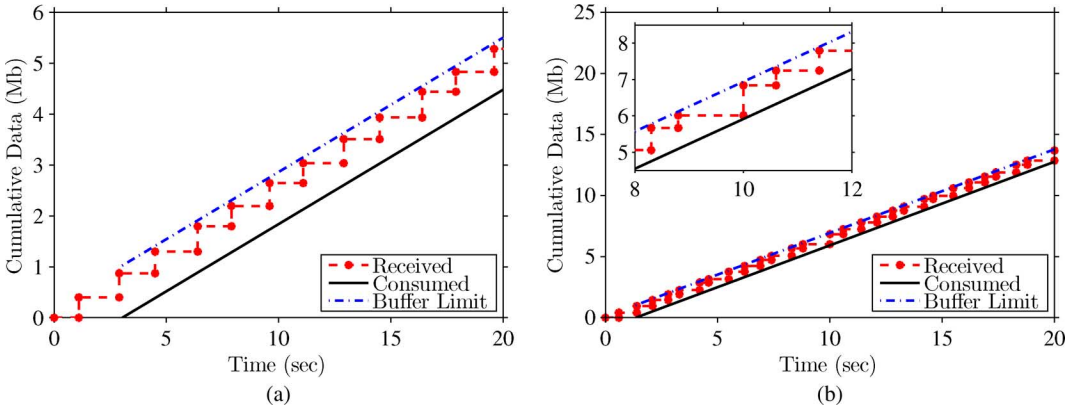


Fig. 10. Buffer level dynamics of the resulting burst schedules of our algorithm: no over/underflow instances are observed. (a) Channel 1, 264 kbps. (b) Channel 4, 684 kbps.

We then implemented our burst scheduling algorithm in the IP encapsulator.

We use Nokia cellular phones, such as N92 and N96 [34], as TV receivers. Although the cellular phones help in assessing the visual quality of videos, they do not provide detailed logging functions of the low-level signals, which are needed to evaluate the scheduling algorithm. To cope with this limitation, we added a DVB-H analyzer [35] to the testbed. This analyzer is attached to a PC via a USB port and provides details on the RF signals and DVB-H channels. It also comes with visualization software that can run on the PC for real-time analysis. More details on the testbed is available online at [36].

For the experiments, we configured the modulator to use a 5-MHz radio channel with quadrature phase-shift keying (QPSK) modulation scheme. According to the DVB-H standard documents, this leads to 5.445 Mbps effective shared bandwidth [15]. We concurrently broadcast 12 TV channels using our algorithm for 10 min. We set the frame length as 10 s. The TV channel bit rates are randomly chosen between 200 and 800 kbps. The receiver buffer size is 1 Mb. For each TV channel, we set up a video streaming server on the base station to send 1-kB IP packets at the chosen bit rate. We set the overhead duration $T_o = 100$ ms. We collect detailed burst logs at the base station. The logs contain the start and end times (in milliseconds) of every burst of data and its size. We developed several software utilities to analyze the logs for three performance metrics: cumulative received bits, time spacing between successive bursts, and energy saving.

### B. Results From the Mobile TV Testbed

*Correctness of the DBS Algorithm:* We first validate the correctness of the proposed algorithm, i.e., we make sure that the algorithm results in no buffer violations for the receivers and no burst conflicts. For buffer violations, we compute the cumulative received bits (from the broadcasting base station) as the time progresses and compare this number against the cumulative consumed bits and the buffer upper limit. The cumulative consumed bits are computed by multiplying the bit rate of each TV channel with the time elapsed. The buffer upper limit is computed as the number of consumed bits plus the receiver's buffer size $Q$, which is set to 1 Mb. Sample results are presented in Fig. 10 for two TV channels; results for other channels are similar. The figure shows the dynamics of the received bits, as the number of bits increases upon receiving a burst and then stays the same till the next burst. Meanwhile, the consumed bits and the buffer upper limits are continuously increasing with slope equals to the bit rate of the TV channel. The figure clearly shows that the curve representing the received bits never goes below the line representing consumed bits (i.e., no buffer underflow instances) and never exceeds the buffer upper limit (i.e., no buffer overflow instances). Note that these two figures show a shorter time period, 20 s, for the clarity of the figure. Nonetheless, this period covers multiple frames and the burst scheduling is identical in successive frames. Thus, the results are the same for the whole streaming period (10 min).

To check for burst conflicts, we compute time spacing between all bursts. We first sort bursts of all TV channels based on their start times. Then, we sequentially compute the time spacing between the start time of a burst and the end time of its immediate, previous, burst. We use the time spacing to validate that the resulting schedule leads to no burst conflicts, as a negative time spacing indicates bursts may intersect with each other. Our logs show that there are no buffer conflicts among bursts computed by our algorithm.

*Energy Saving and Near-Optimality of the DBS Algorithm:* We report the energy saving achieved by receivers of different TV channels when our burst scheduling algorithm is used. Fig. 11(a) shows the energy saving of four representative TV channels; the energy saving of other channels are not shown for the clarity of the figure. We observe that the energy saving for low-bit-rate TV channels can be as high as 92%, while it is only 78% for high-bit-rate TV channels. This significant difference highlights the importance of choosing the appropriate bit rates to encode TV channels carrying diverse video content. The appropriate bit rate is not only important for enhancing the perceived visual quality, but it is also important for maximizing the energy saving and hence prolonging the viewing time on mobile devices.

Next, we compare the energy saving achieved by our algorithm against a very conservative upper bound on the maximum achievable energy saving. Recall that the burst scheduling problem is NP-complete and finding the exact optimal solution

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

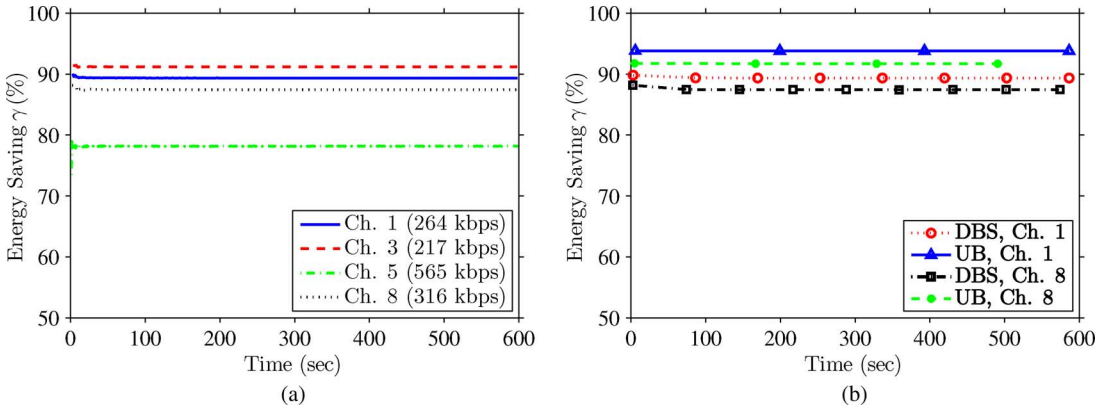


Fig. 11. (a) Energy saving achieved by our algorithm for individual TV channels. (b) Comparing the energy saving achieved by our algorithm against a conservative upper bound on the energy saving.

may take prohibitively long time to compute. We compute this upper bound as follows. For every TV channel, we make the base station broadcast only this channel without any other channels. The base station can maximize the energy saving by allocating the largest burst that can fill the receiver's buffer. The RF circuit of the receiver is then turned off till the data of this burst is consumed. Clearly, this is a conservative upper bound on the energy saving that can be achieved by the receivers of the considered channel. This is because the base station has a complete freedom to allocate the largest burst without considering any interactions from other channels. We repeat this experiment 12 times, once for each considered TV channel. Then, we run our algorithm to compute the burst schedule for the 12 TV channels, and we make the base station broadcast all of them *concurrently*. We compute the energy saving achieved by mobile devices of each TV channel and compare it against the upper bound on the energy saving. We report the results for two sample TV channels in Fig. 11(b); the results for other TV channels are similar. This figure shows that our algorithm produces near-optimal results: The gap between the energy saving achieved by our algorithm and the upper bound is less than 7% in all cases (including the ones not shown in the figure). We emphasize that this gap analysis is very conservative as we compare our algorithm, which concurrently broadcasts several TV channels at arbitrary bit rates, against the maximum energy saving of broadcasting a single TV channel.

*Running Time of the DBS Algorithm:* In all of the above experiments, our algorithm was running in real time on a commodity PC. The running time of our algorithm was in the order of tens of milliseconds. Thus, the algorithm can be invoked frequently as needed and in real time. This is a useful property for the network operators as it allows them to handle the dynamic nature of mobile TV networks and the usual changes in the offered TV programs. For example, broadcasting a commercial ad with high motion and rich visual content (and thus high bit rate) during a talk show (low bit rate) is quite simple: Just before broadcasting the first burst of the commercial ad, our burst scheduling algorithm is invoked to compute a new burst transmission schedule considering the new bit rate of the ad. The same can be done for transitioning between shows and adding new TV channels.

Finally, we mention that the relative start time of each burst is recorded in the header of its predecessor burst such that the receivers know when they need to wake up to receive data [2], [3]. As the start time is sent in the relative form, its accuracy is not affected by any constant delays between the base station and its receivers. However, the start time is sensitive to the clock jitter caused by the inaccuracy of the timers of mobile devices. Therefore, mobile devices cope with this by waking up their RF circuits slightly earlier to absorb the clock inaccuracy, which is referred to as *delay jitter*, and it is in the order of 10 ms [15].

### C. Simulation Setup

We have implemented a simulator for mobile TV broadcast networks in Java. The simulator captures all important aspects relevant to the burst scheduling problem, and it abstracts away details such as sending program guide to mobile devices that are orthogonal to our problem. We developed the simulator to analyze wider ranges of the parameters, including extreme and boundary values that are difficult to exercise in the real testbed. This is useful to fully understand the merits and shortcomings of our algorithm.

Unless otherwise specified, we use the following parameters. The receiver buffer size $Q = 2$ Mb, wireless medium bit rate $R = 6.4$ Mbps, frame length $p = 10$ s, and overhead duration $T_o = 100$ ms. We randomly choose the bit rates of TV channels between 50 to 1000 kbps to emulate different types of TV programs. We repeat each experiment 100 times, and we report the means and 95% confidence intervals of the performance metrics.

We consider several performance metrics, including energy saving $\gamma$, channel switching delay $d$, and running time of our algorithm. We define the channel switching delay as the time a user waits before s/he starts viewing a selected channel when a change of channel is requested by that user. The channel switching delay is an important metric in mobile TV networks, as many users tend to flip through several channels before they decide on a specific channel to watch. The channel switching delay is composed of several accumulated parts, in which the frame refresh delay and time slicing delay are the two dominating contributors [31], [37]. The frame refresh delay refers to the time period between receiving the first bit of a new video

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

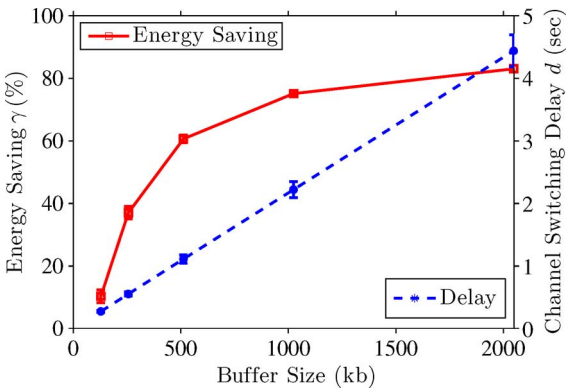


Fig. 12. Impact of the receiver buffer on energy saving and switching delay.

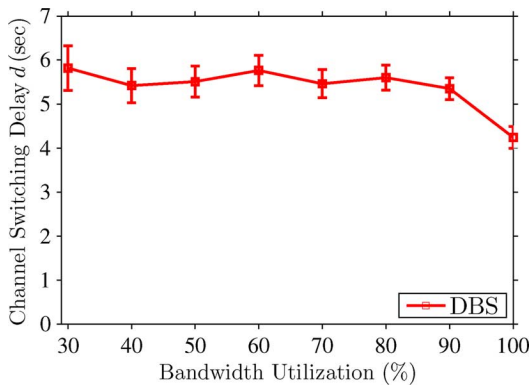


Fig. 13. Impact of bandwidth utilization on switching delay.

stream and receiving the next random access point, typically an intracoded picture, of that video. The time slicing delay refers to the time period between locking on a mobile TV signal and receiving enough bursts of the selected TV channel for a smooth playout. Our simulator captures only the time slicing delay. The frame refresh delay is difficult to simulate because it depends on the specific video content, how it is encoded, and how the frames are organized. In addition, the time slicing delay is a direct outcome of our burst scheduling algorithm, while the frame refresh delay is orthogonal to our algorithm.

### D. Simulation Results

*Tradeoff Between Energy Saving and Channel Switching Delay:* Our results reveal a tradeoff between the achieved energy saving by mobile devices and the average channel switching delay. Furthermore, this tradeoff can be controlled by choosing the appropriate receiver buffer size. To demonstrate this tradeoff, we vary the receiver buffer size between 128 and 2048 kb. We run our simulator and compute the energy saving and the channel switching delay for each buffer value. The results of channel switching delay are shown in Fig. 12. This figure shows that smaller channel switching delays—which are desirable for better viewing experience—require smaller receiver buffer sizes. For example, the average switching delay is reduced from 4 to 2 s by reducing receiver buffer size from 2 to 1 Mb. This indicates that changing the buffer size can control channel switching delays. However, smaller receiver buffer sizes dictate shorter bursts, which increases the energy consumption as RF circuits of the receivers are up more often for smaller bursts and each burst incurs an additional overhead (of at least $T_o$ ms). This is also shown in Fig. 12 as the energy saving diminishes when receiver buffer size becomes very small. The figure indicates that a buffer size of at least 1000 kb is needed to achieve an average energy saving of 75%. With 1000 kb buffer, the average switching delay is about 2 s. If further smaller switching delays are desired without sacrificing the energy saving, other mechanisms may be needed. For example, scalable video coding can be used to encode each TV channel into multiple layers [38]. This scalable video coding, however, imposes an additional overhead as it consumes a small fraction of the wireless medium bandwidth.

*Impact of Wireless Medium Utilization:* Next, we evaluate the performance of our algorithm under different bandwidth utilizations of the shared air medium. We consider various bandwidth utilization from 30% to 100% to cover all practical scenarios. The frame length is fixed at 10 s. For each bandwidth utilization, we consider a burst scheduling problem with TV channels encoded at arbitrary bit rates randomly chosen between 50 and 1000 kbps. We then solve each burst scheduling problem using our DBS algorithm and measure the energy saving and the switching delay. The results indicate that increasing the bandwidth utilization has a minor impact on the energy saving and the switching delay. For example, the average energy saving (figure not shown here due to space limitations) is reduced by less than 5% as the utilization increases from 30% to 100%, while Fig. 13 shows that the channel switching delay decreases from 5.8 to 4.2 s as the utilization increases. This degradation of energy saving is intuitive because heavy loaded mobile TV networks leave smaller rooms for arranging the bursts of different TV channels to save energy, which results in more bursts for each TV channel. More bursts, however, lead to lower channel switching delay because mobile devices can reach the next burst faster. Most importantly, this set of experiments shows that our algorithm is robust and functions properly even in fully loaded networks.

*Impact of Frame Length:* We analyze the impact of various frame lengths on the performance. We vary the frame length from 10 s to 4 min and measure the energy saving and the channel switching delay in each case. We fix the bandwidth utilization at 90%. We report the results in Fig. 14, which shows that increasing the frame length from 10 s to 4 min improves the average energy saving by only about 2%, while it doubles the average channel switching delay. The marginal energy saving improvement is clearly not desirable given the significant increase in the switching delay. This experiment indicates a frame length in the range between 10 and 60 s would achieve a high energy saving without incurring excessive channel switching delays. The specific value of the frame length can be decided by the network operator based on other considerations, such as the desired level of responsiveness to changes in the video content of the TV channels.

*Running Time:* Finally, we report the average running time of our algorithm on a commodity PC with a 2.6-GHz processor that runs Linux. Fig. 15 shows the running time of the algorithm to compute the burst transmission schedules as the bandwidth utilization varies from 30% to 100% while the frame length is

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

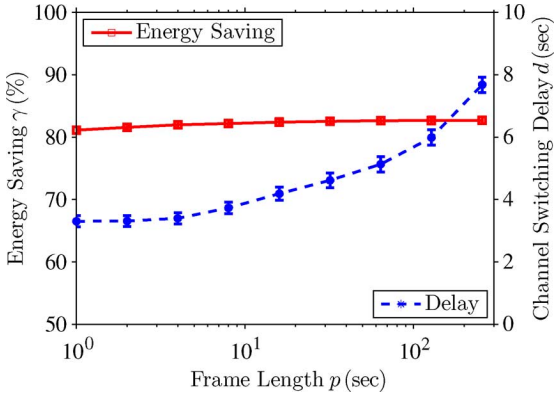


Fig. 14. Impact of frame length on energy saving and switching delay.

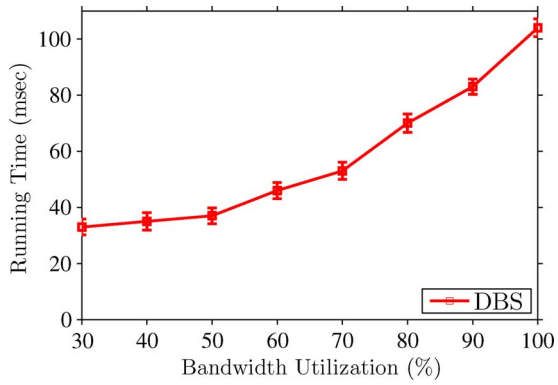


Fig. 15. Running time of our algorithm.

fixed at 10 s. The figure shows that the maximum running time is less than 110 ms on a *commodity* PC, to compute burst schedules for fully utilized mobile TV network. These results confirm our results from the real testbed that our algorithm can indeed run in real time.

We also collect the running time of our algorithm as the frame length varies from 10 to 256 s, while the bandwidth utilization is fixed at 90%. We observe that (figure not shown due to space limitations) for practical frame lengths (less than 60 s), our algorithm terminates in the order of milliseconds. Larger frames contain many bursts and require more computations to carefully specify the start and end of each of them. However, even for such (unusual) large frames, our algorithm terminates in less than 1.1 s. Notice also that the running time of 1.1 s is insignificant compared to a frame length of 256 s because our algorithm is invoked at most once in each frame.

## VI. Conclusion

We studied the energy optimization problem in mobile TV networks. We mathematically formulated the burst scheduling problem for multiple TV channels with arbitrary bit rates. Solving this problem is important because it enables network operators to offer high-quality video programs at bit rates commensurate to the visual complexity of these programs instead of having to use a uniform bit rate for all types of programs, as is currently done in some of the deployed mobile TV networks. The burst scheduling problem is, unfortunately, NP-complete. We proposed a novel approximation algorithm for this burst scheduling problem. We proved that our algorithm has a small approximation factor, and it has a time complexity of $O(pS\log(pS))$, where $S$ is the number of TV channels and $p$ is a constant value for the frame length on which the burst scheduling problem is solved. We also showed how our algorithm can be used to broadcast rate-regulated VBR video streams. We implemented and validated our algorithm in a real mobile TV testbed that complies to the DVB-H standard. We also developed a simulator for mobile TV networks to study the impact of wide ranges of various parameters on the performance of our algorithm. Our experimental and simulation results show that the resulting schedules are correct and the approximation factor of the proposed algorithm is very close to one for most practical mobile TV networks. They also verify that our algorithm can run in real time, and it scales well to large scheduling problems.

## References

[1] *Digital Video Broadcasting-Handheld (DVB-H) home page*, [Online]. Available: http://www.dvb-h.org/

[2] M. Kornfeld and G. May, "DVB-H and IP datacast—Broadcast to handheld devices," *IEEE Trans. Broadcast.*, vol. 53, no. 1, pp. 161–170, Mar. 2007.

[3] *Digital Video Broadcasting (DVB); Transmission System for Handheld Terminals (DVB-H)*, Standard EN 302 304 ver. 1.1.1, European Telecommunications Standards Institute (ETSI), Nov. 2004.

[4] "FLO technology overview," 2009 [Online]. Available: http://www.mediaflo.com/news/pdf/tech_overview.pdf

[5] "Nokia mobile broadcast solution," Feb. 2009 [Online]. Available: http://www.mobiletv.nokia.com/solutions/mbs/

[6] "DVB-H Paris mobile TV customer trial summary page," [Online]. Available: http://www.dvb-h-online.com/Services/services-paris-canalplus.htm

[7] M. Hefeeda and C. Hsu, "On burst transmission scheduling in mobile TV broadcast networks," *IEEE/ACM Trans. Netw.*, Jul. 2009, to be published.

[8] M. Hefeeda and C. Hsu, "Energy optimization in mobile TV broadcast networks," in *Proc. IEEE Innovations*, Al Ain, United Arab Emirates, Dec. 2008, pp. 430–434.

[9] S. Cho, G. Lee, B. Bae, K. Yang, C. Ahn, S. Lee, and C. Ahn, "System and services of Terrestrial Digital Multimedia Broadcasting (T-DMB)," *IEEE Trans. Broadcast.*, vol. 53, no. 1, pp. 171–178, Mar. 2007.

[10] M. Takada and M. Saito, "Transmission system for ISDB-T," *Proc. IEEE*, vol. 94, no. 1, pp. 251–256, Jan. 2006.

[11] G. Faria, J. Henriksson, E. Stare, and P. Talmola, "DVB-H: Digital broadcast services to handheld devices," *Proc. IEEE*, vol. 94, no. 1, pp. 194–209, Jan. 2006.

[12] *Digital Video Broadcasting (DVB); Framing Structure, Channel Coding and Modulation for Digital Terrestrial Television*, Standard EN 300 744 ver. 1.5.1, European Telecommunications Standards Institute (ETSI), Jun. 2004.

[13] K. Iizuka, H. Kawamura, T. Fujiwara, K. Kagoshima, S. Kawama, H. Kijima, M. Koutani, S. Toyoyama, and K. Sakuno, "A 184 mW fully integrated DVB-H tuner with a linearized variable gain LNA and quadrature mixers using cross-coupled transconductor," *IEEE J. Solid-State Circuits*, vol. 42, no. 4, pp. 862–871, Apr. 2007.

[14] S. Pekowsky and K. Maalej, "DVB-H architecture for mobile communications systems," *RF Design Mag.*, pp. 36–42, Apr. 2005.

[15] *Digital Video Broadcasting (DVB); DVB-H Implementation Guidelines*, Standard EN 102 377 ver. 1.3.1, European Telecommunications Standards Institute (ETSI), May 2007.

[16] G. Creus and M. Kuulusa, *Mobile Phone Programming, Optimizing Mobile Software With Built-in Power Profiling*. Dordrecht, The Netherlands: Springer Netherlands, 2007, ch. 25, pp. 449–462.

[17] X. Yang, Y. Song, T. Owens, J. Cosmas, and T. Itagaki, "Performance analysis of time slicing in DVB-H," in *Proc. Joint IST Workshop Mobile Future SympoTIC*, Bratislava, Slovakia, Oct. 2004, pp. 183–186.

[18] E. Balaguer, F. Fitzek, O. Olsen, and M. Gade, "Performance evaluation of power saving strategies for DVB-H services using adaptive MPE-FEC decoding," in *Proc. IEEE PIMRC*, Berlin, Germany, Sep. 2005, pp. 2221–2226.

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.

[19] Q. Zhang, F. Fitzek, and M. Katz, "Cooperative power saving strategies for IP-services supported over DVB-H networks," in *Proc. IEEE WCNC*, Hong Kong, China, Mar. 2007, pp. 4107–4111.
[20] E. Tan, L. Guo, S. Chen, and X. Zhang, "PSM-throttling: Minimizing energy consumption for bulk data communications in WLANs," in *Proc. IEEE ICNP*, Beijing, China, Oct. 2007, pp. 123–132.
[21] C. Hsu and M. Hefeeda, "Time slicing in mobile TV broadcast networks with arbitrary channel bit rates," in *Proc. IEEE INFOCOM*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2231–2239.
[22] P. Brucker, *Scheduling Algorithms*, 4th ed. New York: Springer, 2004.
[23] Y. Bartal, S. Leonardi, G. Shallom, and R. Sitters, "On the value of preemption in scheduling," in *Proc. Workshop APPROX*, Barcelona, Spain, Aug. 2006, pp. 39–48.
[24] O. Braun and G. Schmidt, "Parallel processor scheduling with limited number of preemptions," *SIAM J. Comput.*, vol. 32, no. 3, pp. 671–680, 2003.
[25] R. Motwani, S. Phillips, and E. Torng, "Nonclairvoyant scheduling," *Theor. Comput. Sci.*, vol. 130, no. 1, pp. 17–47, Aug. 1994.
[26] U. Schwiegeishohn, "Preemptive weighted completion time scheduling of parallel jobs," in *Proc. ESA*, Barcelona, Spain, Sep. 1996, pp. 39–51.
[27] T. Lakshman, A. Ortega, and A. Reibman, "VBR video: Tradeoffs and potentials," *Proc. IEEE*, vol. 86, no. 5, pp. 952–973, May 1998.
[28] "Advanced video coding for generic audiovisual services," Joint Video Team, ITU-T Rec. H.264 & ISO/IEC 14496-10 AVC, Nov. 2007.
[29] J. Ribas-Corbera, P. Chou, and S. Regunathan, "A generalized hypothetical reference decoder for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 674–687, Jul. 2003.
[30] *Digital Video Broadcasting (DVB); IP Datacast Over DVB-H: Content Delivery Protocols (CDP) Implementation Guidelines*, Standard EN 102 591 ver. 1.1.1, European Telecommunications Standards Institute (ETSI), Oct. 2007.
[31] M. Rezaei, I. Bouazizi, and M. Gabbouj, "Joint video coding and statistical multiplexing for broadcasting over DVB-H channels," *IEEE Trans. Multimedia*, vol. 10, no. 7, pp. 1455–1464, Dec. 2008.
[32] M. Rezaei, "Video streaming over DVB-H," in *Mobile Multimedia Broadcasting Standards*, F. Luo, Ed. New York: Springer, Nov. 2009, ch. 4, pp. 109–131.
[33] "Dektec DTA-110T PCI modulator," 2008 [Online]. Available: http://www.dektec.com/Products/DTA-110T/
[34] "Nokia Nseries Web site," 2008 [Online]. Available: http://www.nseries.com/
[35] "Divi catch RF-T/H transport stream analyzer," 2008 [Online]. Available: http://www.enensys.com/
[36] M. Hefeeda and C. Hsu, "Design and evaluation of a testbed for mobile TV networks," Simon Fraser University, Tech. Rep. TR 2009-03, Feb. 2009.
[37] M. Rezaei, M. Hannuksela, and M. Gabbouj, "Tune-in time reduction in video streaming over DVB-H," *IEEE Trans. Broadcast.*, vol. 53, no. 1, pp. 320–328, Mar. 2007.
[38] C. Hsu and M. Hefeeda, "Bounding switching delay in mobile TV broadcast networks," in *Proc. ACM/SPIE MMCN*, San Jose, CA, Jan. 2009.

**Cheng-Hsin Hsu** (S'09) received the B.Sc. and M.Sc. degrees from National Chung-Cheng University, Min-Hsiung, Taiwan, in 2000 and 1996, respectively, and the M.Eng. degree from the University of Maryland, College Park in 2003.

He is working toward the Ph.D. degree in the School of Computing Science, Simon Fraser University, Burnaby, BC, Canada. His research interests are in the area of multimedia networking, broadcast networks, wireless networks, and peer-to-peer networks.

**Mohamed M. Hefeeda** (S'01–M'04) received the B.Sc. and M.Sc. degrees from Mansoura University, Mansoura, Egypt, in 1997 and 1994, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 2004.

He is an Assistant Professor with the School of Computing Science, Simon Fraser University, Surrey, BC, Canada, where he leads the Network Systems Lab. His research interests include multimedia networking over wired and wireless networks, peer-to-peer systems, network security, and wireless sensor networks. He has served on more than 20 technical program committees of major conferences in his research areas, including ACM Multimedia, ACM Multimedia Systems, ACM/SPIE Multimedia Computing and Networking (MMCN), IEEE Conference on Network Protocols (ICNP), and IEEE Conference on Communications (ICC). He is an Associate Editor of the *International Journal of Advanced Media and Communication* and the Guest Editor of that journal's special issue on high-quality multimedia streaming in P2P environments. His research on optimizing the performance of mobile multimedia systems has been featured in the *ACM Tech News* (July 1, 2009), CTV British Columbia News (June 26, 2009), and Simon Fraser University's News (May 28, 2009).

Dr. Hefeeda is a Member of the ACM Special Interest Groups on Data Communications (SIGCOMM) and Multimedia (SIGMM). His paper on the hardness of optimally broadcasting multiple video streams with different bit rates won the Best Paper Award in the IEEE Innovations 2008 conference. In addition to publications, he and his students develop actual systems, such as PROMISE, pCache, svcAuth, pCDN, and mobile TV testbed, and contribute the source code to the research community. The mobile TV testbed software developed by his group won the Best Technical Demo Award in the ACM Multimedia 2008 conference. The pCDN (Peer-assisted Content Distribution Network) system is cosponsored by the Canadian Broadcasting Corporation (CBC) and is currently being used to distribute high-quality multimedia content for some of the CBCs Internet streaming services in efficient and cost-effective manner.

Authorized licensed use limited to: SIMON FRASER UNIVERSITY. Downloaded on July 29,2010 at 00:05:28 UTC from IEEE Xplore. Restrictions apply.