This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

Time Slicing in Mobile TV Broadcast Networks with Arbitrary Channel Bit Rates

Cheng-Hsin Hsu
School of Computing Science
Simon Fraser University
Surrey, BC, Canada

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

***Abstract*—Mobile TV networks have received significant attention from the industry and academia, as they have already been deployed in several countries 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 circuit 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 are encoded at arbitrary 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 real implementation in a mobile TV testbed and simulations. Our experimental and simulation results show that the proposed algorithm: (i) is practical and produces correct burst schedules, (ii) achieves near-optimal energy saving for mobile devices, and (iii) runs efficiently in real time.**

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 being tried in many others [1], there is still a large room for optimizing their performance from different angles. This paper addresses one of the most important performance metrics in such networks: minimizing energy consumption for mobile devices while allowing content providers to broadcast diverse video content encoded at arbitrary different 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 DVB-H (Digital Video Broadcast-Handheld) [2] and MediaFLO (Forward Link Only technology) [3], *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 under-flow instances that cause playout glitches and degrade viewing experience for any TV channel.

Current burst scheduling algorithms are simple heuristics. For example, the DVB-H standard documents [4, pp. 66] describe a method to schedule only one TV channel. This method allocates a new burst only after the data of its preceding burst is consumed. To support multiple TV channels using this method, all channels are assumed to be encoded at the same bit rate. Thus bursts belonging to different TV channels will have the same size and can simply be staggered next to each other to create a transmission schedule. This simple method cannot be generalized to multiple TV channels encoded at *different* bit rates. 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. Thus using the simple method may result in many buffer under- and/or over-flow instances, which are not acceptable for a commercial service. Being restricted by the current burst scheduling algorithms, many mobile TV deployments had to resort to encoding all TV channels at the same bit rate. For example, the trial mobile TV service in Paris broadcast 13 TV channels all encoded at 270 kbps [5].

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 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.

In [6], [7], 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

978-1-4244-3513-5/09/$25.00 ©2009 IEEE

paper, we propose a near-optimal algorithm for the general burst scheduling problem that does not require any assumption on the channel bit rates. 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 subscriptions 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. We present theoretical analysis of the proposed algorithm and its approximation factor. We also conduct extensive evaluation of our algorithm using simulations and real implementation in a mobile TV testbed that complies to the DVB-H standard. Our experimental and simulation results show that the proposed algorithm: (i) is practical and produces correct burst schedules without burst conflicts and buffer violation instances, (ii) achieves near-optimal energy saving for mobile devices, and (iii) runs efficiently in real time and scales to large scheduling problems.

The rest of this paper is organized as follows. Section II presents a brief background on mobile TV networks and summarizes the related works in the literature. In Section III, we state the burst scheduling problem in mobile TV systems, and we present the mathematical formulation of the problem. We propose and analyze a new burst scheduling algorithm in Section IV. We present the experimental and simulation results in Section V, and 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 [8], ISDB-T [9], MediaFLO [3], and DVB-H [2], [10]. 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 [3] is a video broadcast system developed by Qualcomm and the FLO forum [11]. The details of the design are not public. In contrast, DVB-H [2], [10] 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 DVB-T (Digital Video Broadcast-Terrestrial) 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 system. Our proposed burst scheduling algorithm resides in the Time Slicing part of the IP Encapsulator, which runs on the base station as shown in Fig. 1.

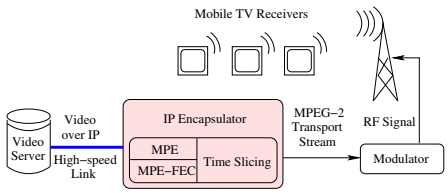


Fig. 1. The main components of mobile TV broadcast networks. Our work optimizes the Time Slicing component.

B. Related Work

Mobile devices have stringent battery capacity and heat dissipation requirements, thus they cannot accommodate mobile TV chips that consume too much energy. For instance, the DVB-H standard [4] specifies that TV signal receivers with power consumption higher than 100 mW cannot be integrated with handheld devices. However, even the state-of-the-art prototype mobile TV chips, such as [13], consume 200 mW, while commercial chips consume more than 500 mW in continuous mode. Therefore, the burst scheduling problem formulated in this paper to minimize the energy consumption of mobile devices is crucial for mobile TV networks.

A number of works have addressed energy saving in mobile TV networks. The authors of [14] and [4] 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 [15] 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 [16] 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. The proposals in [15], [16] 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.

Finally, we note that receivers in mobile TV broadcast networks have separate RF circuits and antennas for TV signals. We only focus on optimizing the energy saving for TV signal receivers. Furthermore, many of the energy saving techniques designed for the general wireless devices are not applicable to mobile TV networks (e.g., [17]), because of the one-way nature of the broadcast networks.

III. Burst Scheduling Problem

A. Problem Statement and Notations

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] and MediaFlo [3]. Each TV channel s, $1 \leq s \leq S$, has a bit rate r_s kbps. We consider a very general problem where r_s can take any value that is less than R. 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]. 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 msec [2], [4].

Let γ_s be the energy saving of the mobile devices receiving TV channel s. γ_s is calculated as the ratio of the time the RF circuits are in off mode to the total time. The γ_s values indicate the energy saving due to time slicing, and it has been used by previous works in the literature [14], [15] and in the standardization documents [4]. We define the average system-wide energy saving over all TV channels as $\gamma = \left(\sum_{s=1}^{S} \gamma_s\right)/S$. 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 networks. In general, longer frame length p provides more chances to shuffle bursts around for better energy saving. However, a longer p also increases the channel switching delay and computation complexity of the burst scheduling algorithm. We will empirically study these tradeoffs in Sec. 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 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 γ. 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: (i) has no data in the buffer to play out (buffer underflow), or (ii) has no space to store data during a burst transmission (buffer overflow).

This burst scheduling problem, as stated above, 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 [6], [7]. We did that by reducing the problem of (non-preemptive) task sequencing with arbitrary release times and deadlines [18, pp. 236] to it.

Note that, the burst scheduling problem might look somewhat similar to preemptive machine scheduling problems at a first glance. However, there is a fundamental difference between our burst scheduling problem and various machine scheduling problems: most machine scheduling problems consider *costless* preemption model [19]. In contrast, our burst scheduling problem adopts *costly* preemption model as each preemption in our problem leads to energy consumption overhead for T_o msec period, which is not negligible compared to the burst size. The costly preemption model has only been considered in a few works [20]–[23]. The authors of [21], [22] partially cope with preemption costs by adding constraints to limit the number of preemptions. The authors of [20] 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 [23] 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, are not applicable to the burst scheduling problem (see [19] for a comprehensive list of machine scheduling algorithms).

B. Problem Formulation

We notice that in this general burst scheduling problem, 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 is no buffer violations. Let n_s be the number of bursts of TV channel s in each frame. To illustrate the receiver buffer dynamics, we demonstrate in Fig. 2 the receiver buffer level of a TV channel with two bursts in each frame. We make two observations on 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 schedule, otherwise the receiver buffer may have over-/under-flow instances.

We denote the start time and burst size of burst k of channel s as f_s^k sec 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 msec before f_s^k and it takes b_s^k/R to transfer b_s^k kb data, the RF on period of burst k for channel s is $[f_s^k - T_o,\ f_s^k + b_s^k/R)$. 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. 2, 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 $\mathbf{L}$, which contains an entry $< n_s, f_s^k, b_s^k, u_s >$ for every TV channel, where $1 \leq s \leq S$.

The burst scheduling problem can be formulated as:

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

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

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

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

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

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

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

The goal is to compute the schedule $\mathbf{L}^*$ to maximize the objective function in (1a), i.e., the system-wide energy saving γ. The constraints (1b)–(1f) guarantee that the resulting burst schedule is feasible as defined in Problem 1. In particular, (1b) ensures that there is no bursts intersection 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. 2. The buffer under- and over-flow 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 u_s.

IV. Near-optimal Algorithm

A. Algorithm Overview

We observe that the hardness of the burst scheduling problem is due to two tightly-coupled constraints: no burst collisions and no receiver buffer violation instances. These two constraints prevent us from employing many techniques used in solving machine scheduling problems. For example, swapping two tasks without compromising feasibility is a common technique in machine scheduling algorithms, but could lead to buffer violation instances in burst scheduling problems. To cope with this hardness, we propose to *decouple* these two constraints. We achieve this by *transforming* the burst scheduling problem into a buffer violation-free problem, which refers to a scheduling problem in which any feasible schedule has no buffer violation instances. As the buffer violation-free problem only has one constraint, similar to machine scheduling problems, it can be solved efficiently with near-optimality. The near-optimal solution for the restricted problem is then transformed back to the original burst scheduling problem, albeit with a small gap from the optimal solution of the original problem due to the transformation. We analytically bound this gap in Sec. IV-B.

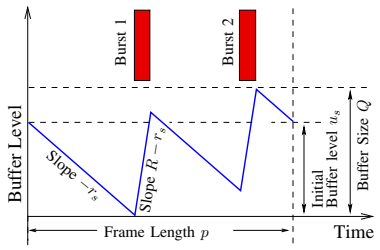


Fig. 2. The dynamics of the buffer size during the burst on and off times.

Based on the decoupling idea, we propose a scheduling algorithm, which we call Double Buffering Scheduling (DBS) algorithm. The DBS algorithm consists of three components: double buffering transform, preemptive burst scheduling, and subframe lateness validation. We describe these components in the following.

Double Buffer Transform. We first explain how we transform a burst scheduling problem into a buffer violation-free problem. The core idea is to utilize *double buffering* in order to get rid of the buffer violation constraint. Double buffering refers to the technique of using two buffers, say B and B', in parallel: B can be drained when B' is filled up, and B' can be drained when B is filled up. Our proposal is to divide the receiver buffer Q into two equal-sized buffers and to divide each frame with length p into several subframes. Mobile devices then use a buffer for receiving (filling up) and another buffer for decoding (draining) in each subframe. Upon reaching a new subframe, mobile devices swap these two buffers: decoding data in the full buffer and saving data into the empty one. Now, if we can schedule the bursts in a way that the number of bits received in the preceding subframe is equal to the number of bits consumed in the current subframe, mobile devices always have data ready to play out, and thus are free from buffer underflow instances. Mobile devices are also free from buffer overflow instances because we explicitly reserve buffer for receiving data in every subframe.

We next precisely define the double buffering transform. For a TV channel s, we divide the frame p into $\lceil 2pr_s/Q \rceil$ subframes, where each subframe (except the last one) has length $Q/2r_s$ sec. We chose such a subframe length because mobile devices of TV channel s consume $Q/2$ kb data in

$Q/2r_s$ sec. Let y_s^k be the burst length that needs to be assigned to subframe k of TV channel s, where $1 \leq s \leq S$ and $1 \leq k \leq \lceil 2pr_s/Q \rceil$. We write y_s^k as:

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

Note that the second row considers the boundary case: the last subframe is shorter if pr_s is not a multiple of $Q/2$. Let x_s^k and z_s^k be the start and end times of subframe k of TV channel s. We write x_s^k and z_s^k as:

$$x_s^k = (k-1)Q/(2r_s), \tag{3}$$

$$z_s^k = \begin{cases} kQ/(2r_s), & 1 \leq k \leq \lfloor 2pr_s/Q \rfloor \ ; \\ p, & k = \lceil 2pr_s/Q \rceil \quad \text{if} \ \ 2pr_s/Q \notin \mathbb{Z}. \end{cases} \tag{4}$$

Also, the second row considers the shorter, last, subframe. Now, we can write $\mathbf{w}$ as the set of all subframes. Our burst scheduling problem in (1) is transformed to another scheduling problem where we schedule one or more bursts to each subframe $w_s^k \in \mathbf{w}$, so that none of the bursts are scheduled before x_s^k, none of the bursts finishes after z_s^k, and the aggregate length of these bursts is y_s^k.

More precisely, we transform our burst scheduling problem into the following buffer violation-free problem:

$$\min \quad \text{number of burst preemptions} \tag{5a}$$

$$\text{s.t.} \quad \text{aggregate length of all scheduled bursts for } s \text{ that fall in subframe } w_s^k \in \mathbf{w} \text{ is } y_s^k. \tag{5b}$$

$$\forall\, 1 \leq s \leq S,\ 1 \leq k \leq \lceil 2pr_s/Q \rceil.$$

This transformed buffer violation-free problem is easier to solve than the burst scheduling problem in Eq. (1). This is because the buffer violation-free property is guaranteed by the transform, which will be proved in Theorem 1.

Preemptive Burst Scheduling. Next, we propose an efficient preemptive burst scheduling algorithm that solves Eq. (5). We first 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 aggregate burst length, i.e., y_s^k. 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., $x_s^k \leq t$. We use outstanding to refer to a subframe that needs more bursts: its current aggregate burst length hasn't met the required aggregate burst length y_s^k. Moreover, we let e_s^k be the completion time of subframe w_s^k, where e_s^k represents the time at which the bursts assigned to w_s^k meet the aggregate burst length y_s^k. Our scheduling algorithm terminates whenever there is no outstanding subframe nor a subframe that has a start time in the future. In Theorem 2, we prove that the preemptive burst scheduling algorithm gives a burst schedule that minimizes the lateness, which is defined as $e_s^k - z_s^k$. The achieved actual lateness allows us to determine whether the resulting burst schedule is feasible for the original burst scheduling problem.

Subframe Lateness Validation. This component checks whether all $e_s^k - z_s^k$ is non-positive. If this is true, we claim that the resulting burst scheduling is feasible for the original burst scheduling problem. Otherwise, the algorithm prompts the network operator to reduce the number of TV channels.

Finally, a high-level pseudo code of the DBS algorithm is given in the technical report [24] for the sake of space limitations.

B. Correctness and Analysis

Due to space limitations, we give all proofs in the technical report [24], which is available online.

In the next lemma, we first show that our algorithm can solve the buffer violation-free problem. The proof is based on transforming any optimal burst schedule to the resulting burst schedule with a finite number of swapping of two bursts without compromising the schedule feasibility. This burst swapping technique is similar to the one used in [19, Theorem 4.4], which solves a single machine preemptive scheduling problem for minimizing task lateness.

Lemma 1: The DBS algorithm finds a burst schedule for the buffer violation-free problem in Eq. (5) if and only if one exists.

We then show, in the next lemma, that the double buffering transform does not affect the existence of feasible schedules. This is proved by showing that the existence of feasible burst schedules for the original problem in Eq. (1) is independent from the receiver buffer size Q.

Lemma 2: There exists a feasible schedule for the buffer violation-free problem in Eq. (5) if and only if there exists a feasible schedule for the original burst scheduling problem in Eq. (1).

With these two lemmas, we prove in the next theorem that the DBS algorithm always finds a feasible burst schedule for the original burst scheduling problem in Eq. (1). This theorem completes our proof of correctness.

Theorem 1 (Correctness): The DBS algorithm produces a feasible schedule for the problem in Eq. (5), which can be transformed to a feasible burst schedule for the original burst scheduling problem in Eq. (1), if one exists.

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

$$\frac{\gamma^*}{\gamma} \leq \frac{1 - T_o R/SQ - 1/S}{1 - 4T_o R/SQ - 1/S}, \tag{6}$$

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

To shed some lights 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

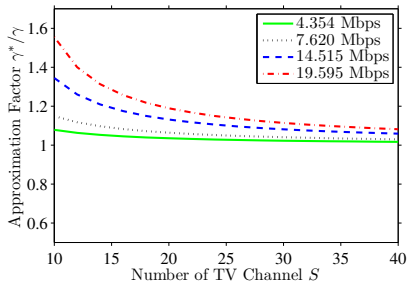


Fig. 3. The approximation factor of the DBS algorithm.

Mbps. These values cover possible bit rates of a 7 MHz band with different modulation and coding schemes [4]. We let the overhead duration $T_o = 0.1$ sec. 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 γ^*/γ and plot it in Fig. 3 for different capacities of the wireless medium. The figure shows that our DBS algorithm produces very close results to the optimal ones, especially for practical values. For example, using our algorithm to broadcast 20 TV channels on 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 one. 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, figures are given in [24], 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.

V. Evaluation

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.

A. Setup of the Mobile TV Testbed

We have set up a complete testbed for mobile TV networks that are based on the common DVB-H standard. The testbed consists of two parts: a base station and receivers, which are described in the following. We use a commodity Linux PC as our base station, in which we installed an RF signal modulator card [25]. This modulator card supports the physical layer of the DVB-H protocol stack and transmits DVB-H signals through a low-power amplifier, which is connected to an antenna. We developed our IP encapsulator based on an open-source project for DVB-H encapsulator [26], which encapsulates IP packets of video streams into MPEG-2 transport streams. The original IP encapsulator only supports burst schedules with uniform inter-burst distance for all TV channels. We have re-designed the time slicing module to be well-structured with clear interfaces in order to facilitate various burst scheduling algorithms. We then implemented our burst scheduling algorithm in the IP encapsulator.

We use Nokia cellular phones, such as N92, 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. Therefore we added the DVB-H Analyzer [27] 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 a visualization software that can run on the PC for real-time analysis.

For the experiments, we configured the modulator to use a 5 MHz radio channel with QPSK (Quadrature Phase-Shift Keying) modulation scheme. According to the DVB-H standard documents, this leads to 5.445 Mbps effective shared bandwidth [4]. We concurrently broadcast 12 TV channels using our algorithm for 10 minutes. We set the frame length as 10 sec. 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 $T_o = 100$ msec. To conduct statistically meaningful performance analysis, we collect detailed burst logs at the base station. The logs contain the start and end times (in msec) 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. In the following subsection, we use these metrics to show that our algorithm produces correct schedules, achieves near optimal energy saving for mobile devices, and is very efficient and can run in real time.

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. A sample result is presented in Fig. 4 for one TV channel, 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

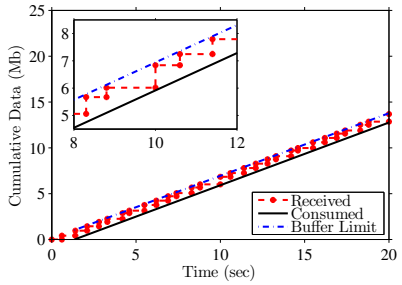


Fig. 4. Buffer dynamics of the resulting burst schedules of our algorithm.

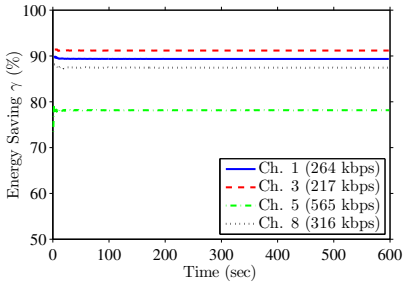


Fig. 5. Energy saving achieved by our algorithm for individual TV channels.

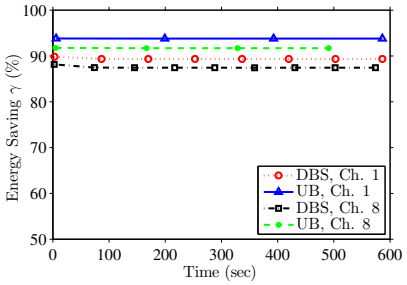


Fig. 6. Comparing the energy saving achieved by our algorithm against a conservative upper bound.

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 this figure shows shorter time period, 20 sec, 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 minutes).

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 the resulting schedule leads to no burst conflict, as a negative time spacing indicates bursts may intersect with each other. Our logs show that there were no buffer conflicts among bursts computed by our algorithm.

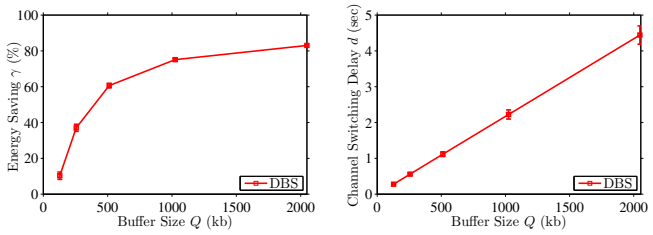
Energy Saving and Near-Optimality of the DBS Algorithm. We report the energy saving achieved by receivers of different TV channels when our burst scheduling algorithm is used. Fig. 5 shows the energy saving of four representative TV channels; the energy saving of other channels are not shown for the 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 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. 6; the results for other TV channels are similar. Fig. 6 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.

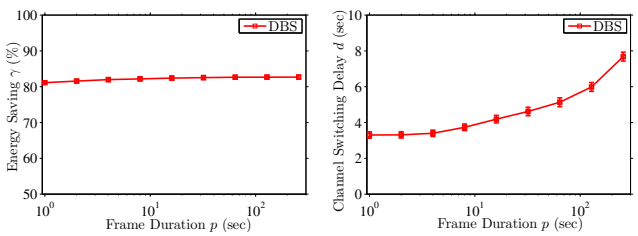
C. Simulation Setup

We have implemented a simulator for mobile TV broadcast networks in Java. The simulator captures all important aspects



(a) (b)

Fig. 7. The impact of changing the receiver buffer size on: (a) energy saving, and (b) channel switching delay.



(a) (b)

Fig. 8. The implication of frame length on: (a) energy saving, and (b) channel switching delay.

relevant to the burst scheduling problem and it abstracts away details, e.g., sending program guide to mobile devices, that are orthogonal to this problem. We developed the simulator to analyze wide 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 $p = 10$ sec, and overhead duration $T_o = 100$ msec. We randomly choose the bit rates of TV channels from 50 to 1,000 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 in all figures.

We consider several performance metrics, including: energy saving γ, channel switching delay d, and running time of our algorithm. The channel switching time 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. 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 composed of several accumulated parts, in which the frame refresh delay and time slicing delay are the two dominating contributors [28], [29]. The frame refresh delay refers to the time period between receiving the first bit of a new video stream and receiving the next random access point, typically an intra-coded 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 play out. 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 trade off 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 kb and 2,048 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. 7(b). The 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 sec 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 the RF circuit of the receivers needs to wake up more often for smaller bursts and in each time it incurs an additional overhead (of at least T_o msec). This is shown in Fig. 7(a) as the energy saving diminishes when receiver buffer size becomes very small. The figure indicates that a buffer size of at least 1,000 kb is needed to achieve an average energy saving of 75%. With 1,000 kb buffer, the average switching delay is about 2 sec.

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 sec. For each bandwidth utilization, we construct burst scheduling problems with TV channels encoded at arbitrary bit rates randomly chosen between 50 and 1,000 kbps. We then solve each burst scheduling problem using our DBS algorithm and measure the energy saving and the switching delay. The results, figures are given in the technical report [24], imply that increasing the bandwidth utilization has a minor impact on the energy saving and the switching delay. For example, the average energy saving is reduced by less than 5% as the utilization increases from 30% to 100%. This degradation is intuitive, because heavy loaded mobile TV networks leave smaller rooms for arranging the bursts of different TV channels to save energy. Nonetheless, 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 sec 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. 8, which shows that increasing the frame length from 10 sec to 4 min only improves the average energy saving by 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 sec would achieve a high energy saving without incurring excessive channel switching delays. The specific value of the frame 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 and runs Linux. Due to space limitations, in [24], we plot the running time of our algorithm to compute the burst schedules as the bandwidth utilization varies from 30% to 100%, while the frame length is fixed at 10 sec. We observe that the maximum running time is less than 110 msec 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.

VI. Conclusions

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 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 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," 2008, http://www.dvb-h.org/.

[2] M. Kornfeld and G. May, "DVB-H and IP Datacast – broadcast to handheld devices," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 161–170, March 2007.

[3] "FLO technology overview," 2008, http://www.qualcomm.com/common/documents/brochures/tech_overview.pdf.

[4] "Digital Video Broadcasting (DVB); DVB-H implementation guidelines. European Telecommunications Standards Institute (ETSI) Standard EN 102 377 Ver. 1.3.1," May 2007.

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

[6] M. Hefeeda and C. Hsu, "On burst transmission scheduling in mobile TV broadcast networks," Simon Fraser University, Tech. Rep., August 2008, available online at http://nsl.cs.sfu.ca/wiki/.

[7] ——, "Energy optimization in mobile TV broadcast networks," in *Proc. of IEEE Innovations in Information Technology (Innovations'08)*, Al Ain, United Arab Emirates, December 2008.

[8] 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 Transactions on Broadcasting*, vol. 53, no. 1, pp. 171–178, March 2007.

[9] M. Takada and M. Saito, "Transmission system for ISDB-T," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 251–256, January 2006.

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

[11] "FLO forum home page," 2008, http://www.floforum.org/.

[12] "Digital Video Broadcasting (DVB); framing structure, channel coding and modulation for digital terrestrial television. European Telecommunications Standards Institute (ETSI) Standard EN 300 744 Ver. 1.5.1," June 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 Journal of Solid-State Circuits*, vol. 42, no. 4, pp. 862–871, April 2007.

[14] X. Yang, Y. Song, T. Owens, J. Cosmas, and T. Itagaki, "Performance analysis of time slicing in DVB-H," in *Proc. of Joint IST Workshop on Mobile Future and Symposium on Trends in Communications (SympoTIC'04)*, Bratislava, Slovakia, October 2004, pp. 183–186.

[15] 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. of IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC'05)*, Berlin, Germany, September 2005, pp. 2221–2226.

[16] Q. Zhang, F. Fitzek, and M. Katz, "Cooperative power saving strategies for IP-services supported over DVB-H networks," in *Proc. of IEEE Wireless Communications and Networking Conference (WCNC'07)*, Hong Kong, China, March 2007, pp. 4107–4111.

[17] E. Tan, L. Guo, S. Chen, and X. Zhang, "PSM-throttling: Minimizing energy consumption for bulk data communications in WLANs," in *Proc. of IEEE International Conference on Network Protocols (ICNP'07)*, Beijing, China, October 2007, pp. 123–132.

[18] M. Garey and D. Johnson, *Computers and Intractability: A Guide to The Theory of NP-completeness*. W. H. Freeman, 1979.

[19] P. Brucker, *Scheduling Algorithms*. Springer, 2004.

[20] Y. Bartal, S. Leonardi, and G. S. R. Sitters, "On the value of preemption in scheduling," in *Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX'06)*, Barcelona, Spain, August 2006, pp. 39–48.

[21] O. Braun and G. Schmidt, "Parallel processor scheduling with limited number of preemptions," *SIAM Journal on Computing*, vol. 32, no. 3, pp. 671–680, 2003.

[22] R. Motwani, S. Phillips, and E. Torng, "Nonclairvoyant scheduling," *Theoretical Computer Science*, vol. 130, no. 1, pp. 17–47, August 1994.

[23] U. Schwiegeishohn, "Preemptive weighted completion time scheduling of parallel jobs," in *Proc. of European Symposium on Algorithms (ESA'96)*, Barcelona, Spain, September 1996, pp. 39–51.

[24] C. Hsu and M. Hefeeda, "Time slicing in mobile TV networks with arbitrary channel bit rates," Simon Fraser University, Tech. Rep., July 2008, available online at http://nsl.cs.sfu.ca/wiki/.

[25] "Dektec modulator," 2008, http://www.dektec.com/Products/DTA-110T.

[26] "FATCAPS project," 2008, http://amuse.ftw.at/downloads/encapsulator.

[27] "DiviCatch analyzer," 2008, http://www.enensys.com.

[28] M. Rezaei, M. Hannuksela, and M. Gabbouj, "Tune-in time reduction in video streaming over DVB-H," *IEEE Transactions on Broadcasting*, vol. 53, no. 1, pp. 320–328, March 2007.

[29] M. Rezaei, I. Bouazizi, and M. Gabbouj, "Joint video coding and statistical multiplexing for broadcasting over DVB-H channels," *IEEE Transactions on Multimedia*, vol. 10, no. 7, pp. 1455–1464, 12 2008.