Dynamic Control of Receiver Buffers in Mobile Video Streaming Systems

Farid Molazem Tabrizi, *Student Member*, *IEEE*, Joseph Peters, and Mohamed Hefeeda, *Senior Member*, *IEEE*

Abstract—We propose a novel algorithm to efficiently transmit multiple Variable-Bit-Rate (VBR) video streams from a base station to mobile receivers in wide-area wireless networks. The algorithm multicasts video streams in bursts to save the energy of mobile devices. In addition, the algorithm adaptively controls the buffer levels of mobile devices receiving different video streams according to the bit rate of the video stream being received by each device. Compared to previous algorithms, the new algorithm enables dynamic control of the wireless channel and allows the base station to transmit more video data on time to mobile receivers. This is done by providing finer control over the bandwidth allocation of the wireless channel. The problem of optimizing energy saving has been shown to be NP-Complete. We prove that our algorithm finds a feasible schedule if one exists and always produces a correct schedule even when dropped frames are unavoidable. We analytically bound the gap between the energy saving resulting from our algorithm and the optimal energy saving and show that our results are close to optimal. We analyze the tradeoff between the fine control over bandwidth allocation and energy saving and demonstrate that in practical situations, flexible and finer control of bandwidth allocation will result in significantly lower frame loss rates while achieving higher energy saving. We have implemented the proposed algorithm as well as two other recent algorithms in a mobile video streaming testbed. Our extensive analysis and results demonstrate that the proposed algorithm outperforms the other two algorithms; it results in higher energy saving for mobile devices and fewer dropped video frames.

Index Terms—Mobile video streaming, mobile TV, energy optimization, quality optimization, wireless video multicasting

1 Introduction

In recent years, mobile devices have become more powerful in terms of computing power, memory, screen size, and screen quality. This provides an improved experience for viewing TV and multimedia content and has resulted in increasing demand for multimedia services for mobile devices [1]. However, video streaming to mobile devices still has many challenges that need to be addressed. For example, mobile devices are small and can only be equipped with small batteries that have limited lifetimes. Thus, conservation of the energy of mobile devices during streaming sessions is needed to prolong the battery lifetime and enable users to watch videos for longer periods. Another challenge for mobile video is the limited wireless bandwidth in wide-area wireless networks. The wireless bandwidth is not only limited, but it is also quite expensive. For instance, Craig Wireless System Ltd. agreed to sell one quarter of its wireless spectrum to a joint venture of Rogers Communication and Bell Canada for $80 million [2], and AT&T sold a 2.5 GHz spectrum to Clearwire Corporation in a $300 million transaction [3]. Thus, for commercially viable mobile video services, network operators should maximize the utilization of their license-based wireless spectrum bands.

In this paper, we consider the problem of multicasting multiple Variable-Bit-Rate (VBR) video streams from a wireless base station to many mobile receivers over a common wireless channel. This problem arises in wide-area wireless networks that offer multimedia content using multicast and broadcast services, such as Advanced Television Systems Committee-Mobile/Handheld (ATSC M/H) [4], WiMAX [5], 3G/4G cellular networks that enable the Multimedia Broadcast Multicast Services (MBMS) [6], and Digital Video Broadcast-Handheld (DVB-H) [7].

We propose a new algorithm to efficiently transmit video streams from a base station to mobile receivers. The transmission of video streams is done in bursts to save the energy of mobile devices. Unlike previous algorithms in the literature, e.g., [8], [9], [10], [11], the new algorithm adaptively controls the buffer levels of mobile devices receiving different video streams according to the bit rates of the video streams. Our algorithm minimizes frame loss by dynamically providing finer control over the allocation of bandwidth to video streams whenever the aggregate bit rate of the video streams increases. In addition, the proposed algorithm uses variable-bit-rate video streams, which are statistically multiplexed to increase the utilization of the expensive wireless bandwidth. We prove that our algorithm finds a feasible schedule for all frames of all video streams if one exists, and always produces a correct schedule. We also analytically bound the gap between the energy saving resulting from our algorithm and the maximum possible energy saving, and we show that our results are close to optimal. We analyze the complexity of our algorithm and show that it can easily run in real time. By dynamically controlling the wireless multicast channel and adjusting the receivers' buffers, our algorithm allows

- *F. Molazem Tabrizi is with the Department of Electrical and Computer Engineering, University of British Columbia, 2332 Main Mall, Vancouver, BC V6T 1Z4, Canada. E-mail: faridm@ece.ubc.ca.*
- *J. Peters and M. Hefeeda are with the School of Computing Science, Simon Fraser University, 250-13450 102nd Ave, Surrey, BC V3T 0A3, Canada. E-mail: {peters, mhefeeda}@cs.sfu.ca.*

Manuscript received 31 May 2011; revised 21 Jan. 2012; accepted 3 Feb. 2012; published online 28 Feb. 2012.
For information on obtaining reprints of this article, please send e-mail to: tmc@computer.org, and reference IEEECS Log Number TMC-2011-05-0286.
Digital Object Identifier no. 10.1109/TMC.2012.56.

the base station to transmit more video data on time to mobile receivers. Also, we discuss the tradeoff between finer control over the wireless channel and the achieved energy saving. We show that in practical scenarios, having more flexible control of the bandwidth in our algorithm results in significantly lower frame loss rates while achieving close to optimal energy saving.

We have implemented the new algorithm in a mobile video streaming testbed [12]. We have also implemented the recent algorithm proposed in [8] and an algorithm used in some commercial base stations for broadcasting VBR video streams [13], [14]. Our empirical results demonstrate the practicality of our new algorithm. Our results also show that our algorithm outperforms other algorithms as it delivers more video frames on time to mobile receivers and it achieves higher energy saving.

The rest of this paper is organized as follows: We review related work in Section 2. In Section 3, we describe the wireless network model that we consider and we state the problem addressed in this paper. We present the proposed algorithm in Section 4. We empirically evaluate our algorithm and compare it against others in Section 5. We conclude the paper in Section 6.

A preliminary version of this work appears in [15]. The current paper presents substantial extensions including an improved algorithm, a complete theoretical analysis, and an extensive empirical evaluation.

2 Related Work

Energy saving at mobile receivers using burst transmission (i.e., time slicing) has been studied in [9] and [10]. Simulations are used in these studies to show that time slicing can improve energy saving for mobile receivers. However, no burst transmission algorithms are presented. Some video encoders adjust the quality of video streams to produce Constant-Bit-Rate (CBR) video streams which makes them less complex when being transmitted. But this results in noticeable quality degradation for the video [16]. A burst transmission algorithm for constant-bit-rate video streams is proposed in [11], but this algorithm cannot handle VBR video streams with fluctuating bit rates. In this paper, we consider the more general VBR video streams which can achieve better visual quality and bandwidth utilization [17].

Variable-bit-rate encoding of video streams results in higher quality video by assigning more bits to complex frames and fewer bits to less complex frames. This results in more complicated requirements for applications of VBR streams [18]. Transmitting VBR video streams over a wireless channel while avoiding buffer overflow and underflow at mobile devices is a difficult problem [19]. Rate smoothing is one approach to reducing the complexity of this problem. The main idea in this approach is to transmit video streams in constant-bit-rate segments and let the receiver store some data ahead of time to avoid large changes in the transmission bit rate of video streams. This reduces the complexities resulting from variability of video stream bit rates. The performance of smoothing algorithms can be measured by different metrics like the peak bit rate, the number of changes in the bit rate, traffic variability, and the receiver buffer requirements. The minimum requirements of rate smoothing algorithms in terms of playback delay, lookahead time, and buffer size are discussed in [20]. The smoothing algorithm proposed in [21] minimizes the number of rate changes.

In [22], a smoothing algorithm is proposed to minimize traffic variability subject to a given receiver buffer size and startup delay. The algorithms in [23] reduce the complexity by controlling the transmission rate of a single video stream to produce a constant-bit-rate stream. Ribas-Corbera et al. [24] suggest smoothing algorithms to reduce the peak bandwidth requirements for video streams. In [25], a Monotonically Decreasing Rate (MDR) algorithm for smoothing the bit rates of video streams is discussed. This algorithm produces segments with monotonically decreasing bit rates for video streams to remove the resource allocation complexities resulting from upward adjustment. The worst case buffer requirement in this algorithm is unbounded which makes it unsuitable for mobile receivers with limited buffer capacity. None of the above smoothing algorithms considers energy saving as a performance metric as these algorithms are not designed for mobile multicast/broadcast networks with limited-energy receivers. In [26], an online Smoothing Algorithm for Bursts (SAB) is introduced which tries to minimize the percentage of lost frames due to bandwidth and buffer limitations. The algorithm calculates the minimum and maximum possible bit rates for video streams without experiencing buffer underflow and overflow instances. SAB transmits video data in bursts to achieve energy saving for mobile receivers, however the algorithm considers the transmission of only one video stream.

A different approach to handling VBR video streams is to employ joint rate control algorithms. In this approach, control of the bit rates of video streams is achieved by using joint video coders [27], [28], [29], which consist of joint rate allocators, VBR coders, and decoders, to jointly encode video streams and dynamically allocate bandwidth to them. For example, Rezaei et al. [27] propose joint video coding and statistical multiplexing for video transmission. In their work, statistical multiplexing is implemented inside the encoders so the bandwidth is distributed among the bitstreams according to their coding complexities. They use time slicing to achieve energy saving. However, this mechanism requires access to expensive joint video encoders.

A recent algorithm for transmitting VBR video streams to mobile devices without requiring joint video encoders is presented in [8]. This algorithm, called SMS, performs statistical multiplexing of video streams. In SMS, the receiver buffer is divided into two equal size parts and the transmission time for each video stream is divided into time windows. During each time window, the base station transmits data to fill half of the buffer while the other half is being drained, and repeats this process in the following time windows. Unlike the proposed algorithm in this paper, the SMS algorithm does not dynamically control the buffers of the receivers. We show that dynamic control of buffers improves the performance in terms of dropped video frames and energy consumption of mobile devices.

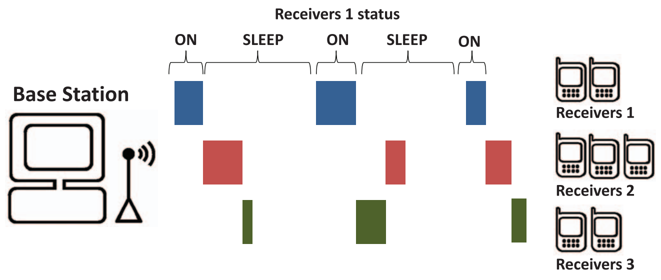


Fig. 1. The network model considered in this paper.

The VBR transmission algorithms deployed in practice are simple heuristics. For example, in the Nokia Mobile Broadcast Solution (MBS) [13], [14], the operator determines a burst size for each video stream and a time interval ΔT based on which bursts are transmitted. This means that for each video stream a burst is transmitted every ΔT seconds. The network operators have to choose the time interval and burst sizes manually and also make sure that their selection does not result in overlapping bursts and buffer overflow/underflow instances for the receivers. The time interval is calculated on the basis of the size of the receiver buffers and the largest bit rate among all video streams and is used for all video streams. Basically, ΔT is set to B/r_S where r_S is the highest bit rate among video streams and B is the buffer size for the receivers. It is difficult to assign bit rate values to VBR video streams to achieve good performance while avoiding buffer underflow and overflow instances at the receivers.

In this paper, we compare our proposed algorithm to the SMS algorithm [8] (which represents the state-of-the-art in the literature) and to the algorithm used in the Nokia Mobile Broadcast Solution (which represents one of the state-of-the-art algorithms in practice).

3 System Model and Problem Statement

3.1 System Model

We study the problem of transmitting several video streams from a wireless base station to a large number of mobile receivers. We focus on multicast and broadcast services enabled in many recent wide-area wireless networks such as DVB-H [7], MediaFLO [30], WiMAX [5], and 3G/4G cellular networks that offer Multimedia Broadcast Multicast Services [6]. In such networks, a portion of the wireless spectrum can be set aside to concurrently broadcast multiple video streams to many mobile receivers. Since the wireless spectrum in wide-area wireless networks is licence-based and expensive, maximizing the utilization of this spectrum is important. To achieve high bandwidth utilization, we employ the variable-bit-rate model for encoding video streams. Unlike the constant-bit-rate model, the VBR model allows statistical multiplexing of video streams [8], and yields better perceived video quality [17]. However, the VBR model makes video transmission much more challenging than the CBR model in mobile video streaming networks [19].

Mobile receivers are typically battery powered. Thus, reducing the energy consumption of mobile receivers is essential. To save energy, we employ the burst transmission model for transmitting video streams, in which the base station transmits the data of each video stream in bursts at a higher bit rate than the bit rate used to encode the video. The burst transmission model allows a mobile device to save energy by turning off its wireless interface between the reception of two bursts [19], [31]. The arrival time of a burst is included in the header of the preceding burst. Thus, the clocks at mobile receivers do not need to be synchronized. In addition, each receiver is assumed to have a buffer to store the received data. Fig. 1 shows a high-level depiction of the system model that we consider. Saving energy of the wireless interface is important because the network interface can consume as much as 70 percent of total power when the display is not on [32], and as much energy as the display when it is on full brightness [33]. In addition, if the wireless interface is on, even if no data is being received, it can consume nearly as much energy as when it is actually receiving data [34].

We note that in wireless broadcast/multicast networks, the parameters of the physical layer, e.g., signal modulation and transmission power, are fixed for all receivers. These parameters are chosen to ensure an acceptable average bit error rate for all receivers in the coverage area of the base station. Therefore, per-user adaptation is typically not used in wireless broadcast/multicast networks. Instead, mechanisms such as Forward Error Correction (FEC) are commonly used to support users in the coverage area.

In broadcast/multicast networks, the transmission is one-way from the base station to the receivers, with no feedback channel from the receivers because there could be too many of them. However, an external channel could exist to allow user interactivity with the transmitted video streams, for example voting for best player during a soccer game. The external channel is a unicast channel. We note that in mobile TV networks, a mobile receiver is typically equipped with a separate interface and controlling logic for receiving mobile TV signals than the interface used to make phone calls and establish unicast sessions. Our work optimizes the downward broadcast/multicast channel, and not the unicast channel.

We list all symbols used in the paper in Table 1.

TABLE 1
Symbols Used in This Paper

Symbol	Description
α	Control parameter indicating the portion of buffer that should be played out before reaching the next control point
B	receiver buffer size (Kb)
b^s_k	Size of burst k of stream s (Kb)
d_s	Playout deadline for receivers of stream s
F	Frame rate of video streams (frames-per-second)
f^s_k	Start time of burst k of stream s (sec)
g_s	Number of frames of stream s that are to be played out between current control point and next one
Γ	Size of scheduling window
γ	Average energy saving among all video streams
h_s	Time of the next control point for video stream s (sec)
I	Total number of frames in a video stream
l^s_i	Size of frame i of stream s (Kb)
M	Maximum burst size (Kb)
m_s	Number of frames sent to receivers of stream s
N	Total number of control points among all video streams
p_s	Number of frames played out at receivers of stream s
R	Available bandwidth (Kbps)
r_s	Average bitrate of video stream s (Kbps)
S	Number of video streams
T	Length of each video stream (sec)
T_o	Wake-up time for the receiver circuit (sec)

3.2 Problem Statement

To achieve the burst transmission of video streams described in Fig. 1, we need to create a transmission schedule that specifies for each stream the number of bursts, the size of each burst, and the start time of each burst. Note that only one burst can be transmitted on the broadcast channel at any time. The problem we address in this section is to design an algorithm to create a transmission schedule for bursts that yields better performance than current algorithms in the literature.

In particular, we study the problem of broadcasting S VBR video streams from a base station to mobile receivers in bursts over a wireless channel of bandwidth R Kbps. The base station runs the transmission scheduling algorithm every Γ sec; we say that Γ is the scheduling window. The base station receives the video data belonging to video streams from streaming servers and/or reads it from local video databases. For example, the base station can be broadcasting a live hockey game, which it receives from the streaming server of a sports media network, as well as several prerecorded TV episodes and movies from a video database. The base station aggregates video data for Γ sec. Then, it computes for each stream s the required number of bursts. We denote the size of burst k of video stream s by b_k^s (Kb), and the transmission start time for it by f_k^s sec. The end time of the transmission for burst k of stream s is $f_k^s + b_k^s/R$ sec.

After computing the schedule, the base station will start transmitting bursts in the next scheduling window. Each burst may contain multiple video frames. We denote the size of frame i of video stream s by l_i^s (Kb). Each video frame i has a decoding deadline, which is i/F, where F is the frame rate (fps). The goals of our scheduling algorithm are 1) maximize the number of frames delivered on time (before their decoding deadlines) for all video streams, and 2) maximize the average energy saving for all mobile receivers. We define the average energy saving as $\gamma = \sum_{s=1}^{S} \gamma_s / S$, where γ_s is the fraction of time that the wireless interfaces of the receivers of stream s are turned off.

4 Proposed Algorithm

4.1 Overview

We propose a novel algorithm, which we call the Adaptive Data Transmission (ADT) algorithm, to solve the burst transmission problem for the VBR video streams described in Section 3. The key idea of the algorithm is to *adaptively* control the buffer levels of mobile devices receiving different video streams.

Since we consider VBR video streams, the bit rate of each video is changing with time according to the visual characteristics of the video. This means that the rate at which the buffer contents of mobile devices are consumed is also changing with time. In other words, there are times when the buffer content of one receiver is being consumed slowly, due to the low bit rate of the video at that time, while the buffer content of another receiver is being consumed at a high rate. Based on this, our algorithm adaptively defines control points in time to decide how much data, and for which stream, the base station should transmit. Our algorithm defines these control points in such a way that when the buffer contents of the receivers are being consumed faster, the decision points are closer together for finer control to ensure that no receiver faces buffer underflow, and when the buffer contents are consumed at a lower rate, the decision points are coarser to save energy. We will show how our algorithm adaptively defines control points and how this saves energy and prevents buffer underflow/overflow instances.

The ADT algorithm defines control points at which it makes decisions about which stream should have access to the wireless medium and for how long it should have access. Control points for each video stream are determined separately based on a parameter α, where $0 < \alpha \leq 1$. This parameter is the fraction of a receiver's buffer capacity B that is played out between two control points. The parameter α can change dynamically between scheduling windows but is the same for all video streams. At a given control point, the base station selects a stream, computes the buffer level of the receivers for the selected stream, and can transmit data as long as there is no buffer overflow at the receivers.

For small values of α, control points are closer to each other (in time) which results in smaller bursts. This gives the base station more flexibility when deciding which video stream should be transmitted to meet its deadline. That is, the base station has more opportunities to adapt to the changing bit rates of the different VBR video streams being transmitted. For example, the base station can quickly transmit more bursts for a video stream experiencing high bit rate in the current scheduling window and fewer bursts for another stream with low bit rate in the current scheduling window. This dynamic adaptation increases the number of video frames that meet their deadlines from the high-bit rate stream while not harming the low-bit rate stream. However, smaller bursts may result in less energy saving for the mobile receivers because they may turn their wireless interfaces on and off more often. In each transition from off to on, the wireless interface incurs an overhead because it has to wake up shortly before the arrival of the burst to initialize its circuits and lock onto the radio frequency of the wireless channel. We denote this overhead by T_o, which is on the order of milliseconds depending on the wireless technology. We analyze the impact of α on the energy saving and number of frames that arrive on time, theoretically in Section 4.3, and empirically using a mobile video streaming testbed in Chapter 5.

4.2 Details

The proposed ADT algorithm is to be run by the wireless base station to schedule the transmission of S video streams to mobile receivers. The algorithm can be called periodically every scheduling window of length Γ sec, and whenever a change in the number of video streams occurs. As will be shown in the next section, the algorithm is computationally efficient and can easily run in real time.

We define several variables that are used in the algorithm. Each video stream s is coded at F fps. We assume that mobile receivers of video streams have a buffer capacity of B Kb. We denote the size of frame i of video stream s by l_i^s (Kb). We denote the time by which the data in the buffer of a receiver of stream s is completely played out (so the buffer is completely drained) by d_s sec.

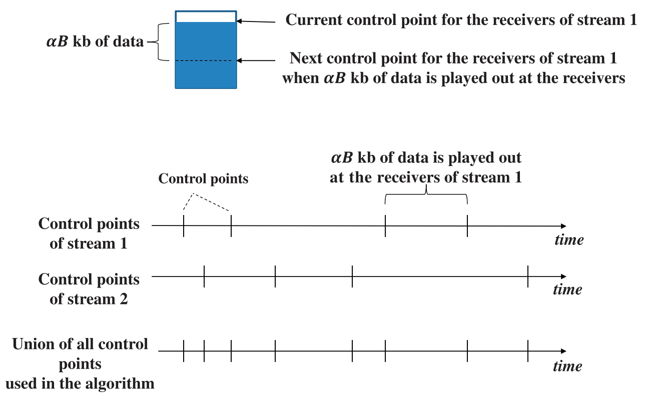

αB kb of data
Current control point for the receivers of stream 1
Next control point for the receivers of stream 1 when αB kb of data is played out at the receivers
Control points
αB kb of data is played out at the receivers of stream 1
Control points of stream 1
Control points of stream 2
Union of all control points used in the algorithm
time

Fig. 2. Control points for two video streams.

Thus, d_s is the deadline for receiving another burst of stream s. We use the parameter M (Kb) to indicate the maximum size of a burst that could be scheduled for a video stream in our algorithm when there are no other limitations like buffer size. In some wireless network standards, there might be limitations on the value of M. In this paper, we assume that the buffer capacity is small, so the upper bound M on the burst size does not affect our algorithm in practice. A control point is a time when the scheduling algorithm decides which stream should be assigned a burst. A control point for video stream s is set every time the receivers of stream s have played out αB Kb of video data as shown in Fig. 2.

The algorithm schedules bursts in scheduling windows of Γ sec. In each scheduling window, the scheduler schedules Γ sec of data that will be played within that scheduling window. We assume that, during each scheduling window, the algorithm also has access to a small amount of data (as large as the size of the buffer) from the next scheduling window. Therefore, if all Γ sec. of video data for the video streams were scheduled within the current scheduling window and there is still some extra time remaining, the scheduler can schedule some bursts from the next scheduling window within the current scheduling window. Let us assume that the algorithm is currently computing the schedule for the time window t_{start} to $t_{start} + \Gamma$ sec. The algorithm defines the variable $t_{schedule}$ and sets it equal to t_{start}. Then, the algorithm computes bursts one by one and keeps incrementing $t_{schedule}$ until it reaches the end of the current scheduling window, i.e., $t_{start} \leq t_{schedule} \leq t_{start} + \Gamma$. For instance, if the algorithm schedules a burst of size 125 Kb on a channel with 1 Mbps bandwidth, then the length of this burst will be 0.125 sec and the algorithm increments $t_{schedule}$ by 0.125 sec. The number of video frames for video stream s that belong to the current scheduling window and are scheduled until $t_{schedule}$ is denoted m_s. This gives the receiver of stream s a playout time of m_s/F sec. Based on this, we can define the playout deadline for stream s at time $t_{schedule}$ as

$$d_s = t_{start} + m_s/F. \tag{1}$$

We let p_s denote the number of frames played out at the receivers of stream s by time $t_{schedule}$. Therefore, we can define p_s as $p_s = t_{schedule} \times F$. The next control point h_s of stream s after time $t_{schedule}$ will be when the receivers of stream s have

Adaptive Data Transmission (ADT) Algorithm

```
     // Input: $S$ VBR video streams
     // Output: Burst schedule for $S$ video streams
1.   compute control points and deadlines for each
1.   stream $s$
2.   while there is a video stream to transmit {
3.     create a new scheduling window from $t_{start}$ to
3.     $t_{start} + \Gamma$
4.     while the current scheduling window is not
4.     complete and $\alpha$ could be updated through
4.     binary search {
5.       pick video stream $s$ having the earliest deadline
6.       schedule a burst until the next control point or
6.       until the buffer is full
7.       if receivers of $s$ are full, do not schedule
7.       any burst for $s$ until the next control point
7.       belonging to $s$
8.       update $t_{schedule}$ based on the length of scheduled
8.       burst
         //gradually increase $\alpha$ if it was reduced in the
         //previous scheduling window
9.       if $\alpha < \alpha_{max}$ and $\alpha$ was not reduced during the
9.       current scheduling window
10.        update $\alpha$ (increase linearly)
11.      update $d_s$ and $h_s$ (control point) for video
11.      stream $s$
12.      if there is a stream $s'$ which is late and $\alpha > \alpha_{min}$
12.      or if $\alpha$ could still be updated through binary
12.      search
           //go back within the scheduling window and
           //reschedule bursts
13.        reset $t_{schedule}$ to $t_{start}$
14.        update $\alpha$ through binary search
15.    }
16.  }
```

Fig. 3. The proposed transmission scheduling algorithm.

played out αB Kb of video data. We compute the number of frames g_s corresponding to this amount as follows:

$$\sum_{i=p_s+1}^{p_s+g_s} l_i^s \leq \alpha B < \sum_{i=p_s+1}^{p_s+g_s+1} l_i^s. \tag{2}$$

The control point h_s is then given by

$$h_s = t_{schedule} + g_s/F. \tag{3}$$

In our algorithm, we use α_{min} and α_{max} to define the range within which the value of α can vary. The operator presets these values based on the desired control over bandwidth and flexibility in energy saving.

The high-level pseudocode of the ADT algorithm is given in Fig. 3. The algorithm works as follows: the scheduler chooses a new video stream to receive a burst at each control point and also at any point when the buffers of the receivers of the currently scheduled burst are full (after scheduling the current burst). When deciding to select a stream, the algorithm finds the stream s' which has the closest deadline $d_{s'}$ and the stream s'' with the closest control point $h_{s''}$. Then the algorithm schedules a burst for stream s' until the control point $h_{s''}$ if this does not exceed the available buffer space at the receivers of s'. Otherwise, the size of the new burst for s' is set to the available buffer space. If the scheduled burst is as large as the available buffer space, which makes the buffer full, then the algorithm does not schedule any bursts for stream s' before it's next control point. This lets the algorithm avoid small bursts for video streams when the buffers are nearly full and results in more energy saving for the receivers. The algorithm repeats the above steps until there are no more bursts to be transmitted from the video streams in the current scheduling window, or until $t_{schedule}$ exceeds $t_{start} + \Gamma$ and there remains data to be transmitted. The latter case means that some frames will not be transmitted. In this case, the algorithm tries to find a better schedule by increasing the number of possible control points to introduce more flexibility. This is done by decreasing the value of α. In order to find the largest value of α that gives us the possibility to transmit all video streams, we do a binary search between α_{min} and α_{max}. The search only considers values between α_{min} and α_{max} with a constant resolution (in our case 0.05), so there is a constant number of choices of values for α. At each step of the binary search, we select a value for α, and run the algorithm. If it was successful, we continue the binary search to try to find a larger α. Otherwise, we continue binary search to choose a smaller α. This process finds the largest value of α for which the algorithm can successfully schedule all video data within the current scheduling window. If no such α is found, then the binary search will result in the selection of α_{min}. It is important to note that the search space is of constant size, so the number of steps in the binary search is also constant, and the scheduling algorithm will be run a constant number of times to find the best α. If α is reduced in a scheduling window, then it will be gradually increased in the following scheduling windows based on a linear function. This means that after scheduling every burst, the value of α will be increased by a constant value. In our work, we have used 0.01 as this constant value.

4.3 Analysis and Complexity of the ADT Algorithm

In this section, we show that the proposed algorithm produces near optimal schedules in terms of energy saving for mobile receivers. We know that the burst scheduling problem is NP-Complete [31]. Thus, solving it optimally and in real time is not feasible. We also show that the proposed algorithm is efficient and that it can run in real time. Finally, we prove that the algorithm results in correct burst transmission schedules with no buffer overflow or underflow instances at the mobile receivers and no two bursts overlapping in time.

The following theorem shows that our algorithm produces near optimal energy saving for mobile devices.

Theorem 1. *The difference between the optimal energy saving and the energy saving of the ADT algorithm is less than or equal to $\frac{\bar{r}T_o}{B}(2/\alpha_{min} - 1)$, where $\bar{r}$ is the average bitrate of all video streams, T_o is the wake-up overhead for the mobile receiver circuits, B is the buffer size, and α_{min} is the minimum value of the control parameter used in the ADT algorithm.*

Proof. The amount of energy saving is related to the number of the bursts scheduled for the video streams. In our algorithm, we schedule a new burst for a new video stream whenever we reach a control point or when we have scheduled a burst for a video stream that completely fills its receivers' buffers. So, the number of bursts will be bounded by the number of control points and the number of times that a receiver buffer gets full. We can see from the algorithm that we will not have a new control point for a video stream s unless we have transmitted at least αB bytes of data since the previous control point for s. Since the value of α can vary, we use $\alpha = \alpha_{min}$ in the remainder of this proof to get bounds. This means that the number of control points is at most $r_sT/(\alpha B)$ where r_s is the average bit rate of video stream s and T (sec) is the total length of the video stream. Note that r_sT is the total number of bits in video stream s. When we schedule a burst that fills the buffer of video stream s, we do not schedule a burst for that stream again before its next control point. Based on this, we can see that between two consecutive control points for stream s, we can have at most one instance that its receiver buffer gets full and consequently, the total number of these instances will be bounded by the total number of control points. Therefore, the total number of bursts for stream s is at most $2r_sT/(\alpha B)$. Hence, the total overhead time that a wireless receiver of stream s is active is at most $2T_or_sT/(\alpha B)$. The wireless receiver of s will also be active during the data transmission time of stream s which is r_sT/R. Consequently, the total time that a wireless receiver of stream s is active is at most $2T_or_sT/(\alpha B) + r_sT/R$. Assuming that all video streams have the same length T, the total time that the wireless receivers of all video streams are active is less than or equal to $2T_o\sum_{s=1}^{S} r_sT/(\alpha B) + \sum_{s=1}^{S} r_sT/R$. Using $\bar{r} = \sum_{s=1}^{S} r_s/S$ as the average bitrate among all video streams, we get a lower bound on the energy saving, γ, of the ADT algorithm:

$$\gamma \geq \frac{\sum_{s=1}^{S} T - 2T_o\sum_{s=1}^{S} r_sT/(\alpha B) - \sum_{s=1}^{S} r_sT/R}{\sum_{s=1}^{S} T}$$
$$= 1 - \bar{r}\left(\frac{2T_o}{\alpha B} + \frac{1}{R}\right).$$

The size of a burst cannot exceed the size of the buffer, so the minimum number of bursts that could be scheduled for video stream s is r_sT/B. Consequently, the overhead time for a wireless receiver of stream s is at least T_or_sT/B and the total time that a wireless receiver of stream s is active is greater than or equal to $T_or_sT/B + r_sT/R$. This gives an upper bound on the optimal energy saving, γ_{opt}:

$$\gamma_{opt} \leq \frac{\sum_{s=1}^{S} T - T_o\sum_{s=1}^{S} r_sT/B - \sum_{s=1}^{S} r_sT/R}{\sum_{s=1}^{S} T}$$
$$= 1 - \bar{r}\left(\frac{T_o}{B} + \frac{1}{R}\right).$$

The difference between the optimal energy saving and the energy saving of the ADT algorithm is $\Delta\gamma = \gamma_{opt} - \gamma \leq \frac{\bar{r}T_o}{B}(2/\alpha_{min} - 1)$. □

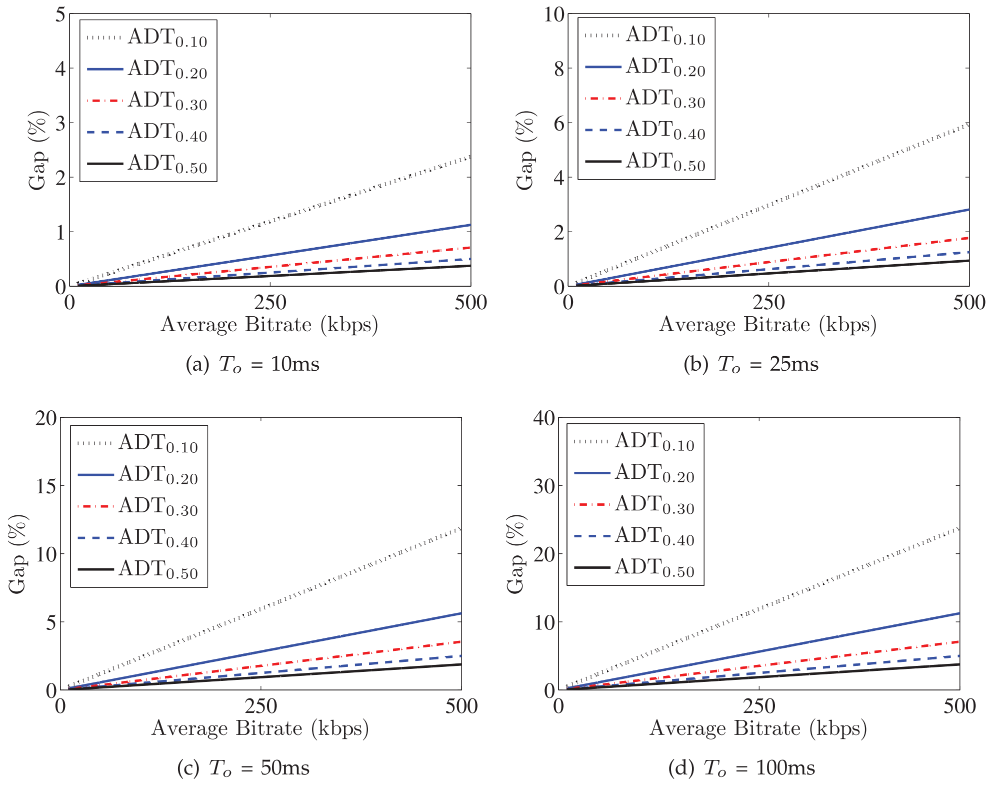


Fig. 4. Maximum possible gap ($\Delta\gamma$) for T_o values of 10 ms, 25 ms, 50 ms, and 100 ms.

Figs. 4a through Fig. 4d show the relationships among $\Delta\gamma$, $\bar{r}$, α, and T_o according to Theorem 1. Each figure plots $\Delta\gamma$ against $\bar{r}$ for fixed values $\alpha = 0.10, 0.20, 0.30, 0.40$, and 0.50 using a fixed value of T_o. The linear relationship of $\Delta\gamma$ with $\bar{r}$ is clear in the figures. It is also clear that decreasing α increases $\Delta\gamma$. The sequence of figures shows that $\Delta\gamma$ increases with increasing wake-up time T_o. Changing the value of α when T_o is small has little impact on $\Delta\gamma$, but when T_o and the average bit rate of the video streams, $\bar{r}$, are both large, the impact of changes in the value of α on $\Delta\gamma$ can be substantial. Similarly, increasing the value of T_o has little impact when α is small, but the impact can be quite large when α and $\bar{r}$ are both large. Therefore, if energy saving is critical, and both T_o and the average bit rate of the video streams are large, the operator can use α as a control parameter to adjust the system for more energy saving or for finer control over the bandwidth which results in a better dropped frame rate.

Next, we show that our proposed algorithm is computationally efficient.

Theorem 2. *The ADT scheduling algorithm runs in time $O(NS)$, where N is the total number of control points and S is the number of video streams.*

Proof. We showed in the proof of Theorem 1 that the number of control points for video stream s is at most $u_s = r_sT/(\alpha B)$ and the number of bursts is at most $2u_s$. Assuming that the length of each video stream is at least αB, each video stream has at least one control point. Let $N = \sum_{s=1}^{S} u_s$ be the total number of control points. At each control point we select the stream with the smallest deadline and schedule a burst for it. Using a priority queue, the stream s with the minimum deadline can be chosen in $\mathrm{O}(1)$ time. Maintaining the current deadline and control point in the priority queue for each stream s takes time $\mathrm{O}(\log S)$ each time that it is changed. Therefore, the total time associated with the priority queue is $\mathrm{O}(N \log S)$. The cost of scheduling a burst depends on the number of frames between two control points. Since the number of bytes in a burst cannot exceed the buffer size B, the number of frames that have to be considered in a scheduling window is at most SB. Since B is a constant, the cost to schedule all bursts is $\mathrm{O}(NS)$ and the time complexity of scheduling is $\mathrm{O}(N \log S + NS) = \mathrm{O}(NS)$. Note that the number of times that the scheduling process is run in a scheduling window is constant because the binary search chooses from among a constant number of α values. Therefore, the total time complexity of the algorithm is $\mathrm{O}(NS)$. □

We note that our algorithm is linear in terms of S (number of streams) and N (number of control points). Thus, our algorithm can easily run in real time. We have actually implemented our algorithm in a mobile video streaming testbed. The algorithm indeed runs in real time on a regular personal computer.

Next, we address the correctness of the proposed algorithm. We assume that $R \geq \sum_{s=1}^{S} r_s$ since a correct schedule without dropped frames is not possible otherwise.

Theorem 3. *The ADT algorithm produces feasible burst schedules with no two bursts overlapping in time and no buffer overflow or underflow instances.*

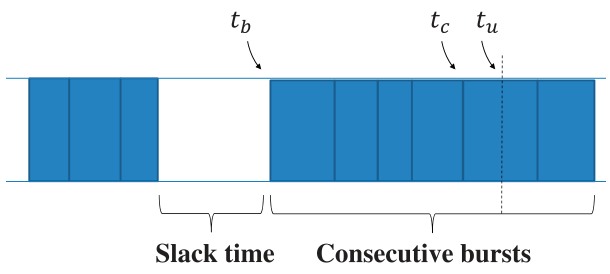


Fig. 5. Bursts are free of buffer underflow instances.

Proof. First, we show that there always exists a value of α that results in no buffer underflow instances. When the ADT algorithm is used, there will be time intervals during which bursts are scheduled, and there will be slack times when no bursts are scheduled (Fig. 5). The slack time could happen for two reasons. The first is that all receiver buffers have been completely filled, so no bursts will be scheduled for them before their next control points. Between two consecutive control points, αB Kb of data will be played out, so the buffer levels of the receivers will be at least $(1-\alpha)B$ during the slack time. The second situation when slack time could occur is at the end of a scheduling window when all video data for the window has been scheduled and there is no more data available. However, we have assumed that there is a small lookahead window that provides access to as much as a buffer of data from the next scheduling window, so this case will never happen and we will not run out of data at the end of a scheduling window. Therefore, when ADT is used, a slack time period can occur for the first reason. Based on this, after a slack time and at the beginning of a period of consecutive bursts, the buffer levels of all receivers will be at least $(1-\alpha)B$ Kb. Now, by way of contradiction, assume that there is a case for which there is no value of α that can avoid buffer underflows. Assume that stream s is the first video stream that will have a buffer underflow in this case. Since all buffer levels are at least $(1-\alpha)B$ Kb during the slack time, buffer underflow for stream s happens during an interval containing consecutive scheduled bursts. Let t_b denote the beginning time of this interval and let t_u denote the time when buffer underflow occurs. Let t_c be the time of the last control point before t_u. Since buffer underflow occurs for stream s, this stream was not selected at time t_c for burst scheduling. Let s' be the stream that was selected at time t_c. Hence, the deadline for stream s' is before the deadline for stream s, which is t_u. Since there is no control point between time t_c and time t_u, and the control points for each stream are at least αB bytes apart, the distance between t_c and t_u is less than αB bytes. It follows that the buffer levels of both streams s and s' are less than αB Kb at time t_c. Now, if we denote the sum of the buffer levels of all video streams at time t by e_t, we have $e_{t_c} < (S-2)B + 2\alpha B$, and $e_{t_b} \geq S(1-\alpha)B$. On the other hand, we have $e_{t_c} = e_{t_b} + R(t_c - t_b) - \sum_{s=1}^{S} r_s(t_c - t_b)$. Since $R \geq \sum_{s=1}^{S} r_s$, we will have $e_{t_c} \geq e_{t_b}$. Therefore, $(S-2)B + 2\alpha B > S(1-\alpha)B$. It follows that $\alpha > 2/(S+2)$, so any value of $\alpha \leq 2/(S+2)$ will avoid buffer underflow. This contradicts the assumption that there is no value of α that avoids buffer underflow.

At each scheduling time, we know the buffer level for each stream s by subtracting the size of the p_s frames that have been played out at the receiver from the total size of the m_s frames that have been scheduled for stream s. Based on this, we know the remaining capacity of the buffer and therefore line 6 of the algorithm (Fig. 3) guarantees that no buffer overflows occur at the receivers.

Finally, we show that no two scheduled bursts overlap in time. Each new burst is created starting at the current scheduling time $t_{schedule}$ in line 6 of the algorithm (Fig. 3). After scheduling a burst in line 6, $t_{schedule}$ is updated in line 8 and moved forward as much as the length of the burst scheduled in line 6. Other than line 8, there is only one place where $t_{schedule}$ is updated in the algorithm and that is line 13. In line 13, $t_{schedule}$ is reset to the start time and the scheduling is started from scratch. Therefore, the ADT algorithm produces a feasible burst schedule for the base station. □

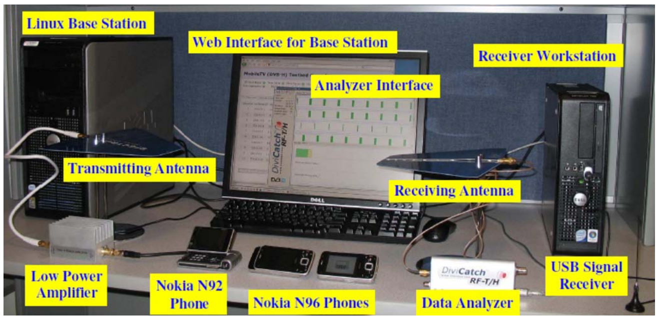


Fig. 6. The testbed used in the evaluation.

Theorem 3 shows that the ADT algorithm will find a correct schedule if one exists and there are no constraints on the value chosen for α. If $R < \sum_{s=1}^{S} r_s$, then the available bandwidth of the wireless channel is insufficient to transmit all frames of all video streams and buffer underflows are inevitable. Buffer undeflows are also possible if α_{min} is set too high or the distance d_α between the values of α considered by the binary search is too large. In particular, the algorithm must be able to find a value $a \leq 2/(S+2)$ to avoid underflows. If $2/(S+2) < \alpha_{min} + d_\alpha$, then ADT will not find α and underflows could occur. However, even if undeflows occur, the schedules produced by ADT are correct with no buffer overflows and no pair of bursts overlapping. Furthermore, the values α_{min} and d_α are under the control of the operator.

5 Evaluation

In this section, we rigorously analyze the performance of the proposed algorithm and compare its performance to recent algorithms. We start by describing the setup of our mobile video testbed. Then, we present detailed results for different performance metrics.

5.1 Testbed and Setup

The testbed that we used to evaluate our algorithm, shown in Fig. 6, consists of a base station, mobile receivers, and

TABLE 2
Video Streams Used in the Experiments

Bit rate (Kbps)	Description
83 – 121	8 video streams, documentary, movie, news
240 – 260	3 video streams, demo video and advertisements
400 – 600	6 video streams, movie, sport, news

data analyzers. The base station includes an RF signal modulator which produces DVB-H standard-compliant signals. The signals are amplified to around 0 dB before transmission through a low-cost antenna to provide coverage to approximately 20 m for cellular phones. The base station has a multithreaded design that enables the broadcast of multiple video streams in real time on a low-end server.

The mobile receivers in our testbed are Nokia N96 cell phones that have DVB-H signal receivers and video players. Two DVB-H analyzers are included in the testbed system. The first one, a DiviCatch RF T/H tester [35], has a graphical interface for monitoring detailed information about the received signals, including burst schedules and burst jitters. The second analyzer, a dvbSAM [36], is used to access and analyze received data at the byte level to monitor the correctness of the received content.

We implemented our ADT algorithm, the SMS algorithm [8], and the Nokia Mobile Broadcast Solution [13], [14], in the testbed and integrated them with the IP encapsulator of the transmitter. We set the overhead T_o to 100 msec and the modulator to a Quadrature Phase Shift Keying (QPSK) scheme and a 5 MHz radio channel. According to DVB-H standard documents, this gives us net bandwidth of 5.18 Mbps. We fixed the maximum receiver buffer size B at 4 Mb (500 KB). We prepared a test set of 17 diverse VBR video streams to evaluate the algorithms. The different content of the streams (TV commercials, sports, action movies, documentaries) provided a wide range of video characteristics with average bit rates ranging from 25 Kbps to 600 Kbps. The average bit rate of the video streams was 260 Kbps. Each video stream was played at 30 fps and had length 566 sec. Information about the video streams is summarized in Table 2. We transmitted the 17 VBR video streams concurrently to the receivers and we collect detailed statistics from the analyzers. Each experiment was repeated for each of the three algorithms (ADT, SMS, and MBS). In addition, for our ADT algorithm, we repeated the experiments with several values of the buffer adaptation parameter α.

5.2 Results for Dropped Video Frames

Dropped frames are frames that are received at the mobile receivers after their decoding deadlines or not received at all. The number of dropped frames is an important quality of service metric as it impacts the visual quality and smoothness of the received videos. Fig. 7 shows the cumulative total over all video streams of the numbers of dropped frames for ADT with fixed values of α ranging from 0.10 to 0.50, and for SMS and MBS. The figure clearly shows that ADT consistently drops significantly fewer frames than the SMS and MBS algorithms. The figure also shows the effect of decreasing the value of α for the ADT algorithm. The total number of dropped frames decreases from 6,630 with $\alpha = 0.50$ to 2,074 with $\alpha = 0.20$. No frames are dropped when α is reduced to 0.10. On the other hand, the SMS algorithm is significantly worse with 9,605 dropped frames. The results for MBS in Fig. 7 were obtained by running the algorithm for each video stream with different assigned bit rates ranging from 0.25 times the average bit rate to four times the average bit rate of the video stream and then choosing the best result for each video stream. Even in this total of best cases, the number of dropped frames is more than 34,000. In practice, an operator heuristically chooses the assigned bit rates for video streams, so the results in practice likely will be worse.

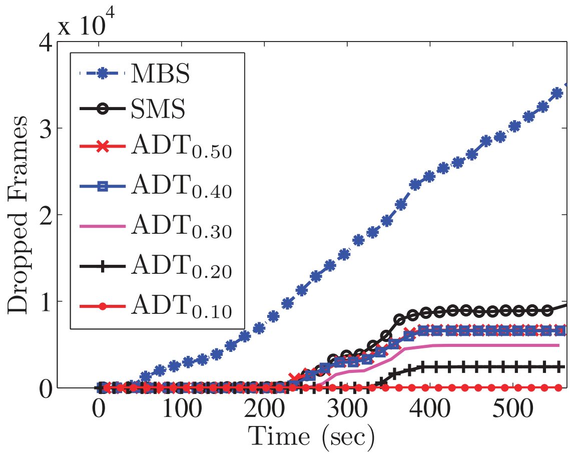


Fig. 7. Total number of dropped video frames.

We counted the number of dropped frames for each video stream to check whether the ADT algorithm improves the quality of some video streams at the expense of others. Two samples of our results are shown in Fig. 8; others are similar for different values of α. The curves in the figure show the average over all streams of the percentage of dropped frames; each point on the curves is the average percentage of frames transmitted to that point in time that were dropped. The bars show the ranges over all video streams. As shown in the figure, the difference between the maximum and minimum dropped frame percentages at the end of the transmission period is small. We conclude that the ADT algorithm does not sacrifice the quality of service for some streams to achieve good aggregate results.

We further analyze the patterns of dropped video frames for each algorithm in more detail. We plot the total number of dropped frames during each 1 sec interval across all video streams. Our results are shown in Fig. 9. For the ADT algorithm with $\alpha = 0.50$, frames were dropped mostly during a 150 sec period (between 220 and 370 sec) because several video streams have high bit rate during this period. Reducing α to 0.20 permits finer control over the distribution of bandwidth among the video streams and significantly fewer frames were dropped as shown in Fig. 9d. Further reducing α to 0.10 eliminated all dropped frames as we have already seen in Fig. 7. The SMS algorithm on the other hand dropped up to 42 percent of the frames during

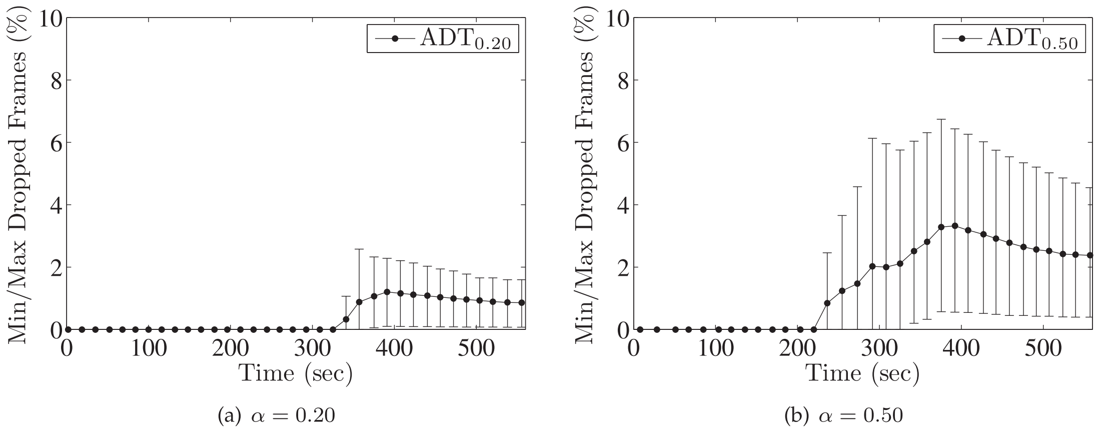


Fig. 8. Minimum and maximum number of dropped video frames.

the period in which the aggregate bit rate of all streams is high. Using the MBS algorithm results in dropping frames in a very wide time range. The results from these experiments confirm the performance benefits that our ADT algorithm achieves by dynamically adapting to the changing bit rate nature of VBR video streams by controlling the level of receivers' buffers through the parameter α.

We present the aggregate of average bit rate of all video streams over time in Fig. 10. We also show the aggregate of instantaneous bit rate of video streams in Fig. 11. Although the aggregate of the average bit rate over time does not exceed the bandwidth capacity, the aggregate of the instantaneous bit rate of the streams exceeds available bandwidth. We can see the effect of such changes on the average buffer level of the receivers in Fig. 12. When the instantaneous bit rate of the video streams exceeds the available bandwidth, the average buffer level of receivers decreases. This is because the size of the played out data exceeds transmitted data. But when the available bandwidth exceeds the aggregate of the instantaneous bit rate, the average buffer level of the receivers increases.

5.3 Results for Energy Saving

We compute the average energy saving γ achieved across all video streams, which represents the average amount of time that the wireless interfaces are off. The results are shown in Fig. 13. The figure shows that the average saving resulting from the ADT algorithm when $\alpha = 0.5$ is very close to the optimal energy saving, and is about 6.74 percent more than the average energy saving achieved by the SMS algorithm. Also, our experiments show that the energy saving achieved by ADT is considerably higher than the MBS algorithm, which achieves an average energy saving of up to 41.14 percent.

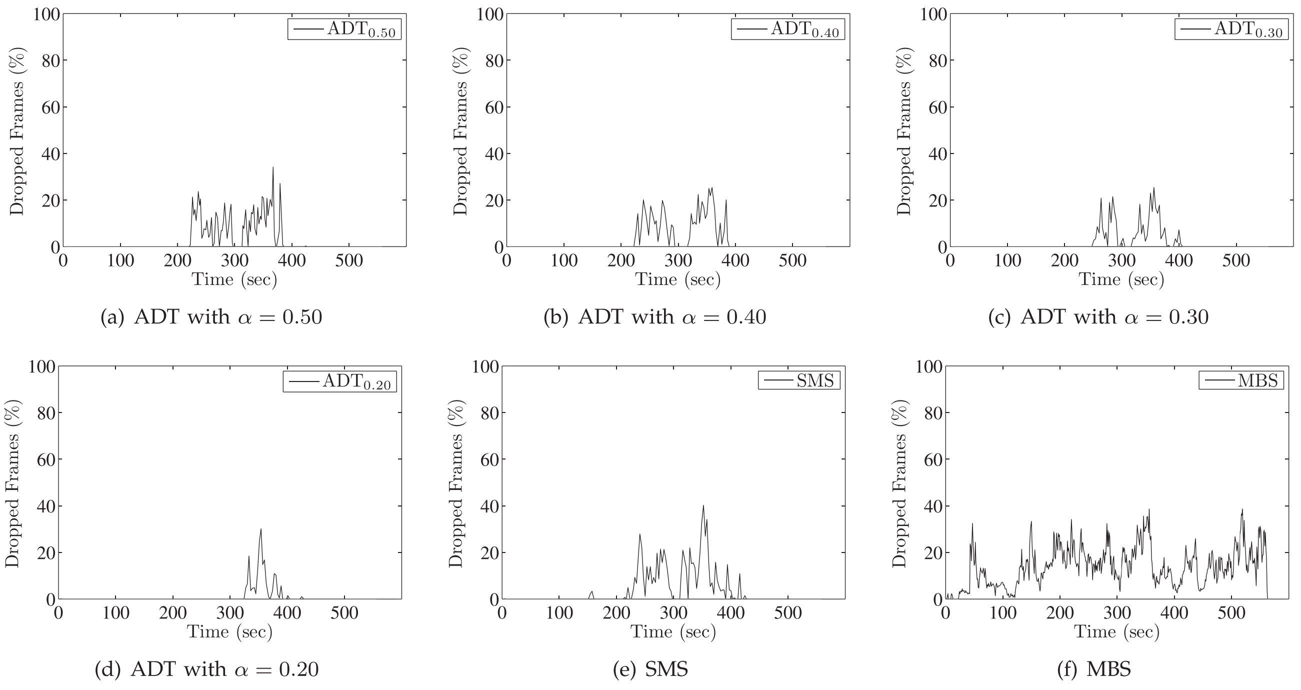


Fig. 9. Dropped video frames over 1 sec periods.

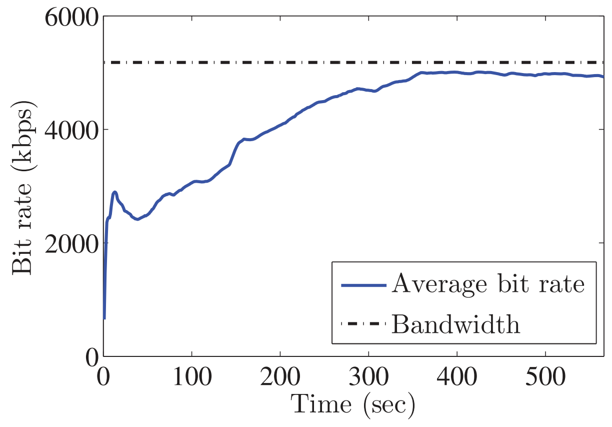


Fig. 10. Aggregate of average bit rate of all video streams.

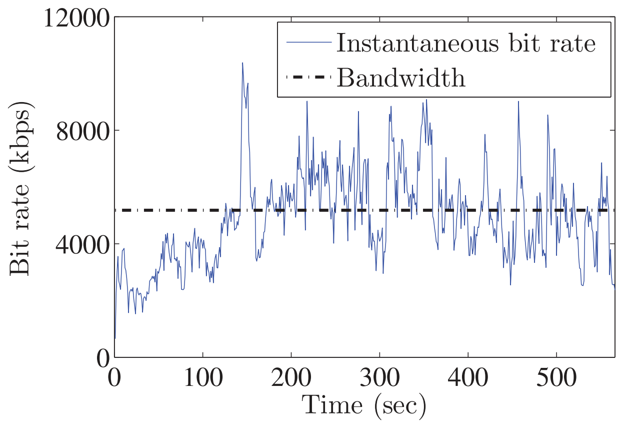


Fig. 11. Aggregate of instantaneous bit rate of all video streams.

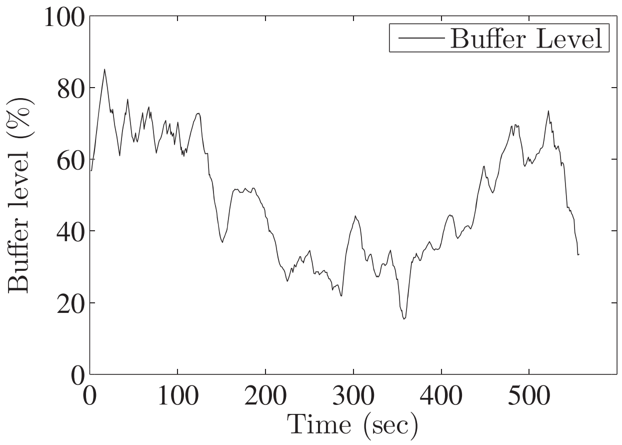


Fig. 12. Average buffer level of receivers.

The impact of changing α on the energy saving achieved by the ADT algorithm is shown in Fig. 14. The average over all video streams of the energy saving is 89.52 percent for the ADT algorithm when $\alpha = 0.10$ which is better than the SMS algorithm. Increasing α to 0.50 increases the energy saving to 93.47 percent. The small improvement of 3.95 percent in energy saving is nontrivial but it might not be large enough in many practical situations to offset the advantage of minimizing the number of dropped frames by setting $\alpha = 0.10$.

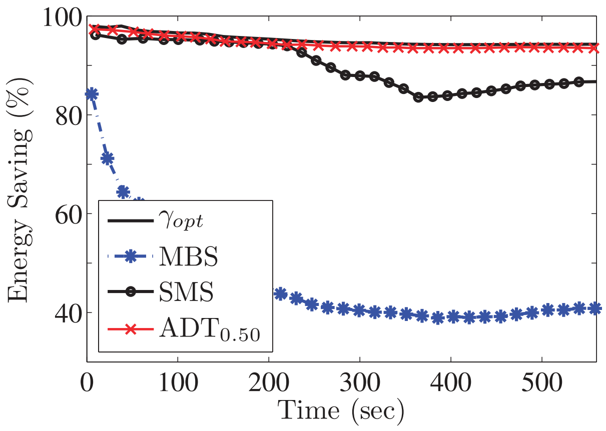


Fig. 13. Average energy saving.

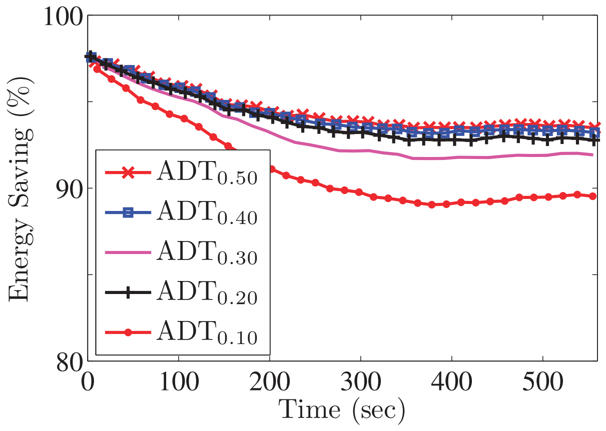


Fig. 14. Average energy saving with different values of α.

Next, we compare the energy saving resulting from the ADT algorithm to the upper and lower bounds from Theorem 1. Fig. 15 shows the results for α values of 0.5, 0.3, and 0.1. It is important to note that the upper bound in the figure is not the optimal energy saving, but it is a conservative upper bound on the optimal energy saving which may not be achievable. According to the figure, the gap between our algorithm and the conservative upper bound is less than 5 percent for $\alpha = 0.5$.

Finally, we measured the energy saving for the receivers of each individual stream to check whether the ADT algorithm unfairly saves more energy for some receivers at the expense of others. A sample of our results is shown in Fig. 16. The figure confirms that ADT does not sacrifice energy saving in some streams to achieve good average energy saving.

In summary, the results in this section show that the proposed algorithm achieves higher energy saving than previous algorithms, that the saving is uniform across all mobile receivers, and that the saving is very close to the optimal energy saving.

5.4 Impact of Changing α

The results in the previous two sections indicated that changing the value of the buffer control parameter α impacts the number of dropped frames and the energy saving achieved by mobile devices. In this section, we analyze this impact for different values of α. We vary the

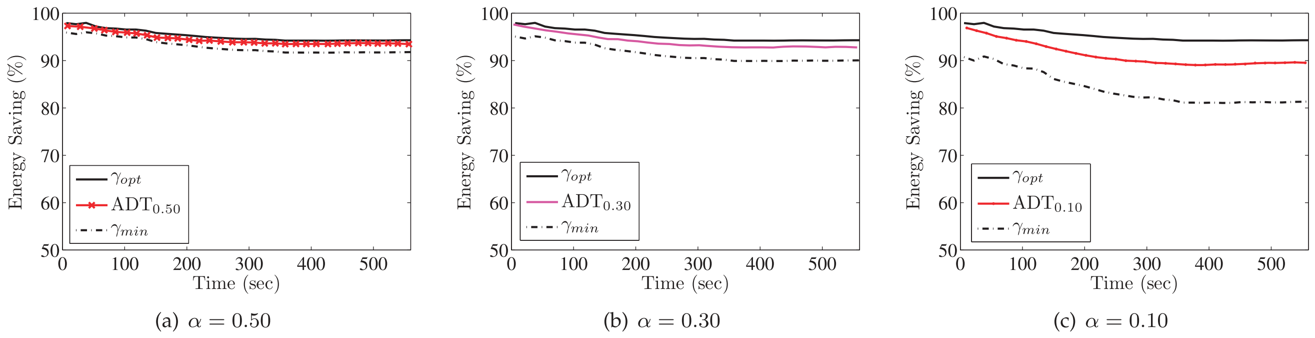


Fig. 15. Upper and lower bounds for energy saving.

value of α between 0.10 and 0.50 for our ADT algorithm. For each value of α, we ran the experiment of transmitting all video streams and measured the average energy saving as well as the percentage of video frames that arrived on time. We plot the results in Fig. 17. Notice that the figure has two y-axes. The results show that α exposes a tradeoff between the energy saving and the percentage of frames that arrive on time. Small α values allow finer control of the wireless channel and thus increase the chance of transmitting video frames on time, but small values of α result in more bursts being used to transmit the same amount of video data, and each burst incurs an overhead of T_o. Thus, small α values result in smaller energy saving. Based on this, an operator can use larger values of α for lower bit rate video streams and smaller values of α to have finer control over the bandwidth of high bit rate video streams. It is important to note that even with the smallest value $\alpha = 0.10$ that we tested, our ADT algorithm still achieves energy saving as high as the state-of-the-art SMS algorithm.

5.5 Dynamic Adjustment of α

The previous section shows the existence of a tradeoff between energy saving and the number of on-time video frames, which is controlled by α. Our ADT algorithm is capable of adaptively updating the value of α at run time based on bandwidth limitations and changes in the bit rates of the video streams to give a good balance between the number of dropped frames and energy saving. This is illustrated in the pseudocode in Fig. 3, where the ADT algorithm dynamically adjusts the value of α. The ADT algorithm tries to maintain the largest value of α that achieves the minimum number of dropped frames because larger values of α allow greater energy saving.

We conducted an experiment to show the merits of dynamically adjusting the value of α in the range of 0.1 to 0.5, instead of fixing it as in the previous sections. We ran our ADT algorithm and allowed it to change α at run time. We chose different values for the scheduling window Γ: 30, 60, and 120 sec. The results are shown in Fig. 18. When the scheduling window is 30 sec, the energy saving converges to 92 percent (Fig. 18b) which is approximately the same as with a fixed value of $\alpha = 0.3$, but the total number of dropped frames for each video stream is 3,162 (Fig. 18a) which is much better than the 4,896 dropped frames with fixed $\alpha = 0.3$. When the scheduling window is 60 sec, the energy saving is 91.6 percent which is similar to the energy saving results with a fixed value of $\alpha = 0.2$ but the total number of dropped frames for each streams is 1,496 which is much better than 2,414 with fixed $\alpha = 0.2$. With a scheduling window of 120 sec, no frames are dropped and the energy saving converges to 90.51 percent which is better than the energy saving of 89.52 percent that results with a fixed value of $\alpha = 0.1$ (which also resulted in no dropped frames).

In summary, the results of this experiment show that dynamically choosing the parameter α as done by our algorithm achieves both good energy saving and small numbers of dropped video frames by dynamically using smaller values of α when we need finer control over

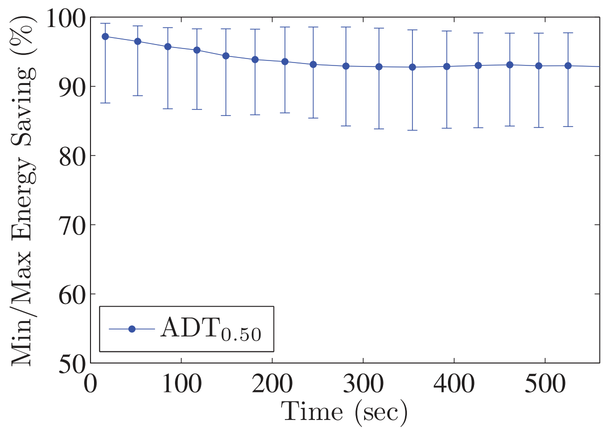


Fig. 16. Min and max energy saving.

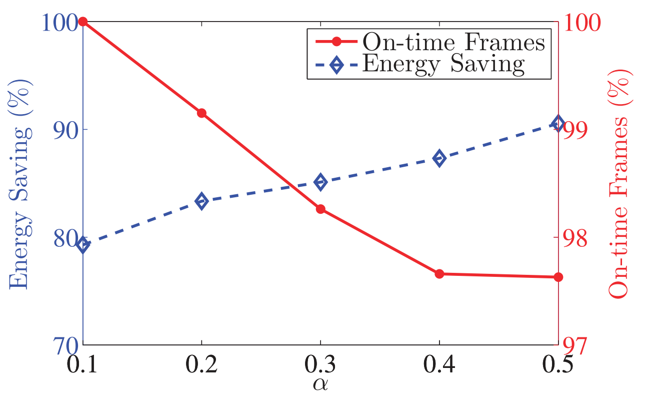
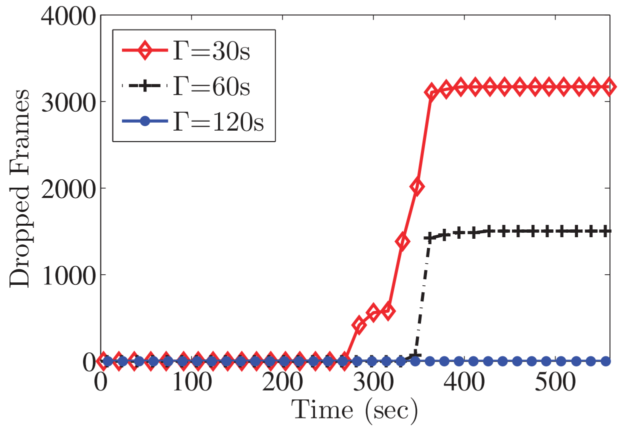
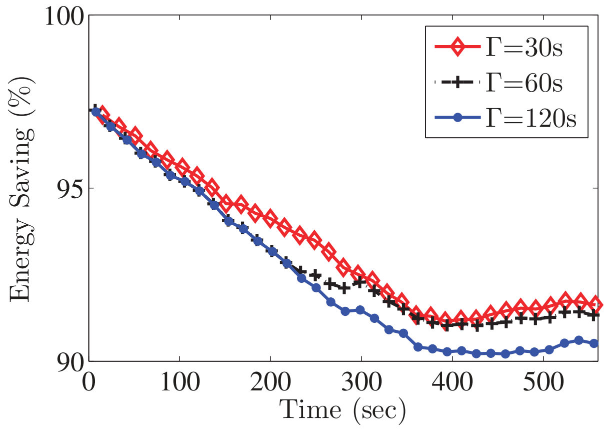


Fig. 17. The impact of α on energy saving and number of dropped video frames.



(a) Dropped frames



(b) Energy saving

Fig. 18. The impact of dynamically choosing α.

bandwidth, and larger values of α when we do not have bandwidth limitations. More importantly, network operators do not need to heuristically set the parameter α for our ADT algorithm.

6 Conclusions

We have presented a new algorithm for transmitting multiple VBR video streams to energy-constrained mobile devices. The algorithm is to be used by base stations in wide-area wireless networks that offer multimedia services in broadcast/multicast modes such as the Multimedia Broadcast Multicast Service of 3G/4G cellular networks, WiMAX networks, and Digital Video Broadcast-Handheld networks. One of the novel aspects of the proposed algorithm is its ability to dynamically adjust the level of receivers' buffers according to the bit rates of the video streams being received by each receiver. This is done through dynamically controlling the allocation of wireless bandwidth to video streams. We showed that the proposed algorithm computes feasible schedules whenever they exist. We analytically computed the gap between energy saving resulting from our algorithm and the optimal energy saving, and we showed that this gap is small. We presented a proof-of-concept implementation of the proposed algorithm in a mobile video streaming testbed. We compared the new algorithm against other algorithms proposed in the literature and used in practice. We conducted extensive empirical analysis using a number of VBR video streams with diverse visual characteristics and bit rates. Our results show that the proposed algorithm yields high energy saving for mobile receivers and reduces the number of video frames that miss their deadlines. The results also demonstrate that the proposed algorithm outperforms the current state-of-the-art algorithms.

Acknowledgments

This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada and in part by the British Columbia Innovation Council.

References

[1] "Global IPTV Market Analysis (2006-2010)," http://www.rncos.com/Report/IM063.htm, 2006.

[2] "Craig Wireless to Sell Canadian Spectrum for $80m," http://www.cbc.ca/fp/story/2010/03/26/2729450.html. 2010.

[3] "AT&T Sells Wireless Spectrum in Southeast to Clearwire Corporation," http://www.att.com/gen/press-room?pid=4800&cdvn=news&newsarticleid=23428, 2007.

[4] "ATSC Mobile DTV Standard," http://www.openmobilevideo.com/about-mobile-dtv/standards, 2009.

[5] *IEEE 802.16: Broadband Wireless Metropolitan Area Network,* http://standards.ieee.org/getieee802/802.16.html, 2009.

[6] S. Parkvall, E. Englund, M. Lundevall, and J. Torsner, "Evolving 3G Mobile Systems: Broadband and Broadcast Services in WCDMA," *IEEE Comm. Magazine,* vol. 44, no. 2, pp. 30-36, Feb. 2006.

[7] M. Kornfeld and G. May, "DVB-H and IP Datacast - Broadcast to Handheld Devices," *IEEE Trans. Broadcasting,* vol. 53, no. 1, pp. 161-170, Mar. 2007.

[8] C. Hsu and M. Hefeeda, "On Statistical Multiplexing of Variable-Bit-Rate Video Streams in Mobile Systems," *Proc. 17th ACM Int'l Conf. Multimedia (Multimedia '09),* pp. 411-420, Oct. 2009.

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

[10] X. Yang, Y. Song, T. Owens, J. Cosmas, and T. Itagaki, "Performance Analysis of Time Slicing in DVB-H," *Proc. Joint IST Workshop Mobile Future and Symp. Trends in Comm. (SympoTIC '04),* pp. 183-186, Oct. 2004.

[11] C. Hsu and M. Hefeeda, "Time Slicing in Mobile TV Broadcast Networks with Arbitrary Channel Bit Rates," *Proc. IEEE INFOCOM,* pp. 2231-2239, Apr. 2009.

[12] M. Hefeeda, C. Hsu, and Y. Liu, "Testbed and Experiments for Mobile TV (DVB-H) Networks," *Proc. 16th ACM Int'l Conf. Multimedia (Multimedia '08),* Oct. 2008.

[13] "Nokia Announces World's First Commercial Solution for Managing DVB-H Broadcast Services," http://press.nokia.com/2005/10/31/nokia-announces-worlds-first-commercial-solution-for-managing-dvb-h-broadcast-services, 2005.

[14] Private Communication with Nokia's Engineers Managing Mobile TV Base Stations, 2010.

[15] F. Molazem Tabrizi, J. Peters, and M. Hefeeda, "Adaptive Transmission of Variable-Bit-Rate Video Streams to Mobile Devices," *Proc. 10th Int'l IFIP TC 6 Conf. Networking (NETWORKING '11),* 2011.

[16] W. Feng and J. Rexford, "Performance Evaluation of Smoothing Algorithms for Transmitting Prerecorded Variable-Bit-Rate Video," *IEEE Trans. Multimedia,* vol. 1, no. 3, pp. 302-312, Sept. 1999.

[17] T. Lakshman, A. Ortega, and A. Reibman, "VBR Video: Tradeoffs and Potentials," *Proc. IEEE,* vol. 86, no. 5, pp. 952-973, May 1998.

[18] M. Grossglauser, S. Keshav, and D.N.C. Tse, "RCBR: A Simple and Efficient Service for Multiple Time-Scale Traffic," *IEEE/ACM Trans. Networking,* vol. 5, no. 6, pp. 741-755, Dec. 1997.

[19] M. Rezaei, "Video Streaming over DVB-H," *Mobile Multimedia Broadcasting Standards,* F. Luo, ed., pp. 109-131, Springer, Nov. 2009.

[20] P. Thiran, J. yves Le Boudec, and F. Worm, "Network Calculus Applied to Optimal Multimedia Smoothing," *Proc. IEEE INFOCOM,* pp. 1474-1483, Apr. 2001.

[21] J. Zhang and J. Hui, "Applying Traffic Smoothing Techniques for Quality of Service Control in VBR Video Transmissions," *Computer Comm.,* vol. 21, no. 4, pp. 375-389, 1998.

[22] J.D. Salehi, S.-L. Zhang, J. Kurose, and D. Towsley, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements through Optimal Smoothing," *IEEE/ACM Trans. Networking,* vol. 6, no. 4, pp. 397-410, Aug. 1998.

[23] J. Lin, R. Chang, J. Ho, and F. Lai, "FOS: A Funnel-Based Approach for Optimal Online Traffic Smoothing of Live Video," *IEEE Trans. Multimedia,* vol. 8, no. 5, pp. 996-1004, Oct. 2006.

[24] J. Ribas-Corbera, P. Chou, and S. Regunathan, "A Generalized Hypothetical Reference Decoder for H.264/AVC," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 13, no. 7, pp. 674-687, July 2003.

[25] H. Lai, J. Lee, and L. Chen, "A Monotonic-Decreasing Rate Scheduler for Variable-Bit-Rate Video Streaming," *IEEE Trans. Circuits and Systems for Video Technology,* vol. 15, no. 2, pp. 221-231, Feb. 2005.

[26] P. Camarda, G. Tommaso, and D. Striccoli, "A Smoothing Algorithm for Time Slicing DVB-H Video Transmission with Bandwidth Constraints," *Proc. ACM Int'l Mobile Multimedia Comm. Conf. (MobiMedia '06),* Sept. 2006.

[27] 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.

[28] Z. He and D. Wu, "Linear Rate Control and Optimum Statistical Multiplexing for H.264 Video Broadcast," *IEEE Trans. Multimedia,* vol. 10, no. 7, pp. 1237-1249, Nov. 2008.

[29] M. Jacobs, J. Barbarien, S. Tondeur, R.V. de Walle, T. Paridaens, and P. Schelkens, "Statistical Multiplexing Using SVC," *Proc. IEEE Int'l Symp. Broadband Multimedia Systems and Broadcasting (BMSB '08),* pp. 1-6, Mar. 2008.

[30] "FLO Technology Overview," http://www.mediaflo.com/news/pdf/tech_overview.pdf, 2009.

[31] M. Hefeeda and C. Hsu, "On Burst Transmission Scheduling in Mobile TV Broadcast Networks," *IEEE/ACM Trans. Networking,* vol. 18, no. 2, pp. 610-623, Apr. 2010.

[32] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces," *Proc. ACM MobiSys,* pp. 220-232, 2006.

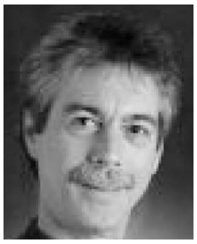
[33] A. Carroll and G. Heiser, "An Analysis of Power Consumption in a Smartphone," *Proc. Usenix Technical Conf.,* pp. 1-14, June 2010.

[34] S. Chandra, "Wireless Network Interface Energy Consumption: Implications for Popular Streaming Formats," *Multimedia Systems,* vol. 9, pp. 185-201, Aug. 2003.

[35] "DiviCatch RF-T/H Transport Stream Analyzer," http://www.enensys.com, 2008.

[36] "dvbSAM DVB-H Solution for Analysis, Monitoring, and Measurement," http://www.decontis.com, 2008.

Farid Molazem Tabrizi received the BSc degree in software engineering from the Amirkabir University of Technology, Iran, in 2005, and the MSc degree in computing science from Simon Fraser University, Canada, in 2011. He is currently a PhD candidate in the Department of Electrical and Computer Engineering, University of British Columbia, Canada. His research interests include distributed systems and multimedia networking. He is a student member of the IEEE.



Joseph Peters received the BSc degree from the University of Waterloo and the MSc and PhD degrees from the University of Toronto, Canada, the latter in 1983. He is a professor in the School of Computing Science, Simon Fraser University, Canada. His research interests include multimedia networking over wireless networks and the modeling and performance analysis of networks including ad hoc, sensor, peer-to-peer, radio, and wired networks.

Mohamed Hefeeda received the BSc and MSc degrees from Mansoura University, Egypt, in 1994 and 1997, respectively, and the PhD degree from Purdue University in 2004. He is an associate professor in the School of Computing Science, Simon Fraser University, Canada, where he leads the Network Systems Lab. His research interests include multimedia networking over wired and wireless networks, peer-to-peer systems, mobile multimedia, and Internet protocols. Dr. Hefeeda won the Best Paper Award at the IEEE Innovations 2008 conference for his paper on the hardness of optimally broadcasting multiple video streams with different bitrates. In addition to publications, he and his students have developed actual systems, such as pCache, svcAuth, pCDN, and mobile TV testbed. The mobile TV testbed software developed by his group won the Best Technical Demo Award at the ACM Multimedia 2008 conference. He serves as the preservation editor of the *ACM Special Interest Group on Multimedia (SIGMM) Web Magazine*. He served as the program chair of the ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2010) and as a program cochair of the International Conference on Multimedia and Expo (ICME 2011). In addition, he has served on many technical program committees of major conferences in his research areas, including ACM Multimedia, ACM Multimedia Systems, and the IEEE Conference on Network Protocols (ICNP). He is on the editorial boards of the *ACM Transactions on Multimedia Computing*, *Communications and Applications* (ACM TOMCCAP), the *Journal of Multimedia*, and the *International Journal of Advanced Media and Communication*. He is a senior member of the IEEE.