Energy-Efficient Multicasting of Scalable Video Streams Over WiMAX Networks

Somsubhra Sharangi, Ramesh Krishnamurti, and Mohamed Hefeeda, *Senior Member, IEEE*

***Abstract*—The Multicast/Broadcast Service (MBS) feature of mobile WiMAX network is a promising technology for providing wireless multimedia, because it allows the delivery of multimedia content to large-scale user communities in a cost-efficient manner. In this paper, we consider WiMAX networks that transmit multiple video streams encoded in scalable manner to mobile receivers using the MBS feature. We focus on two research problems in such networks: 1) maximizing the video quality and 2) minimizing energy consumption for mobile receivers. We formulate and solve the substream selection problem to maximize the video quality, which arises when multiple scalable video streams are broadcast to mobile receivers with limited resources. We show that this problem is NP-Complete, and design a polynomial time approximation algorithm to solve it. We prove that the solutions computed by our algorithm are always within a small constant factor from the optimal solutions. In addition, we extend our algorithm to reduce the energy consumption of mobile receivers. This is done by transmitting the selected substreams in bursts, which allows mobile receivers to turn off their wireless interfaces to save energy. We show how our algorithm constructs burst transmission schedules that reduce energy consumption without sacrificing the video quality. Using extensive simulation and mathematical analysis, we show that the proposed algorithm: 1) is efficient in terms of execution time, 2) achieves high radio resource utilization, 3) maximizes the received video quality, and 4) minimizes the energy consumption for mobile receivers.**

***Index Terms*—Energy efficiency, mobile multimedia, scalable video coding, video streaming, WiMAX, wireless scheduling.**

I. Introduction

THE demand for mobile multimedia streams has been increasing in the past few years as indicated by multiple market analysis studies [1], [2]. Multimedia streams can be delivered to mobile devices over different wireless networks, including 3G, WiFi, and WiMAX networks. In this paper, we focus on multimedia streaming over WiMAX networks, which are specified by the IEEE 802.16 standard [3]. Although the currently deployed WiMAX networks are mostly used to provide wireless Internet access to subscribers, the WiMAX standard supports various network services. One of these services is the Multicast and Broadcast Service (MBS), which can be used to deliver multimedia traffic to large-scale user communities. For example, Yota Telecom [4] has recently started a mobile TV service with 25 channels over its 10 Mbps mobile WiMAX network, and UDCast [5] has announced plans for developing broadcast TV service supporting around 50 channels over mobile WiMAX. It is expected that more WiMAX deployments will offer mobile multimedia services in the near future. Although considerable amount of work has been done to make these deployments feasible, several research problems remain to be addressed in order to optimize the quality of the offered multimedia services.

In this paper, we address two important problems in multimedia streaming over WiMAX networks: 1) maximizing the video quality and 2) minimizing energy consumption for mobile receivers. In particular, we consider broadcasting multiple scalable video streams to mobile receivers. A scalable video stream is composed of multiple layers, where each layer improves the spatial, temporal, or the visual quality of the rendered video to the user. Because of their flexibility, scalable video streams can efficiently support heterogeneous receivers, adapt to network conditions, and utilize the available wireless bandwidth. We mathematically formulate the problem of selecting the best set of substreams (or layers) from the scalable video streams in order to maximize the quality for mobile receivers. We show that this problem is NP-Complete. Thus, optimally solving it in real time may not be computationally feasible. We propose an approximation algorithm that produces near-optimal solutions and runs in real time. We analytically show that the approximation factor is close to one.

In addition, since many subscribers of the WiMAX multimedia services are expected to be mobile users with energy-constrained devices such as smart phones, minimizing the energy consumption of these devices becomes an important problem in order to extend the viewing time. To address this problem, we extend our algorithm to reduce the energy consumption of mobile receivers. The extended algorithm first selects the best substreams and then transmits these substreams in bursts. The burst transmission of the video data enables mobile receivers to turn off their wireless interfaces for longer periods of time in order to save energy. Our algorithm carefully constructs the burst transmission schedules that reduce the energy consumption without sacrificing the video quality or introducing any buffer overflow or underflow instances. We rigorously evaluate the proposed algorithm using simulation and mathematical analysis. Our results show that the proposed algorithm can efficiently run in real time, achieves high utilization of the wireless bandwidth, minimizes the energy consumption for mobile receivers, and maximizes the video quality.

The rest of this paper is organized as follows. In Section II, we present a brief background on video streaming over WiMAX

Manuscript received May 21, 2010; revised August 24, 2010; accepted August 27, 2010. Date of publication September 16, 2010; date of current version January 19, 2011. 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. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Thinh Nguyen.

The authors are with the School of Computing Science, Simon Fraser University, Surrey, BC V3T 0A3, Canada (e-mail: ssa121a@cs.sfu.ca; ramesh@cs.sfu.ca; mhefeeda@cs.sfu.ca).

Digital Object Identifier 10.1109/TMM.2010.2076799

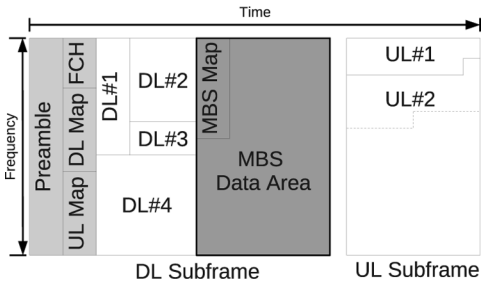

Fig. 1. Frame structure in WiMAX.

networks, and we summarize the related works in the literature. We state the substream selection problem and present the analytical formulation for it in Section III. In Section IV, we present the proposed approximation algorithm to efficiently solve the substream selection problem. In Section V, we present our extended algorithm which reduces the energy consumption for mobile receivers. Section VI describes our simulation setup and results. Finally, we conclude the paper in Section VII. A preliminary version of this paper appeared in [6].

II. Background and Related Work

A. Brief Background

A video streaming service over WiMAX networks is composed of three main entities: 1) content source, 2) WiMAX base station, and 3) WiMAX subscribers. Content sources are national TV broadcasters, local broadcasters, Internet TV operations, and other video broadcast service providers. Multimedia contents are aggregated from different sources and sent to the WiMAX base station. The WiMAX base station constructs a schedule to transmit the incoming data to the subscribers.

In the WiMAX physical layer, data are transmitted over multiple carriers in time division duplex (TDD) frames. As illustrated in Fig. 1, each frame contains header information and upload/download maps followed by bursts of user data. Since video dissemination is expected to be a prevalent traffic pattern in future networks, the WiMAX standard defines a service called MBS in the MAC layer to facilitate broadcast and multicast. Using MBS, a certain area in each TDD frame can be set aside for multicast-only or broadcast-only data, as shown in Fig. 1. The entire frame can also be designated as a download-only broadcast frame. A major task of the MBS module is to allocate video data from multiple streams to the MBS data area in each frame such that the real-time nature of all video streams is maintained. Further, the allocation algorithm must consider that the receiver devices have limited buffer capacity which may cause data loss due to buffer overflow. These constraints impose stringent QoS and efficiency demands on the allocation algorithm. In the first part of this paper, we present an algorithm to select the best subset of scalable video streams and allocate them in the MBS area.

The WiMAX standard has defined different sleep mode operations to facilitate power conservation for mobile subscribers. In the sleep mode operation, the base station informs the mobile subscriber about the sleep interval, during which the mobile subscriber switches off its RF circuitry. To utilize the energy conservation mechanism of the sleep mode, the data are sent out in bursts, instead of continuous transmission. After receiving a burst of data for a short time, receivers go into sleep mode for a pre-computed period of time. Once we compute a transmission schedule, the sleep intervals for different streams are embedded in the last frame of a burst. At the start of transmission, each mobile subscriber receives the first burst and comes to know about its sleep and active intervals. It then accordingly switches its receiver on or off to receive only the relevant frames. This procedure is continuously repeated for each scheduling window. In the second part of this paper, we design a burst transmission algorithm to reduce the energy consumption of mobile devices.

B. Related Work

1) Video Streaming Over WiMAX: Wang *et al.* [7] discuss an architecture for video broadcasting in a multi-base-station WiMAX system. Their work focuses on coverage and spectral efficiency issues and considers only temporal video scalability. Cohen *et al.* [8] combine a group of TDD frames together into a super-frame. They describe a cost-based scheme where a cost function is associated with each user-channel pair. Three user interaction models are considered: 1) user can either be statically hooked to a channel, 2) user can choose to listen to a channel, or 3) the user channel association can keep changing based on the transmission medium conditions. The work in [8] does not consider the delay requirements which are central to video streaming. Hosein [9] describes the frame allocation problem for broadcasting variable bit rate video over WiMAX, but does not consider scalable video content.

Jiang *et al.* [10] propose a scheme to transmit scalable video streams in which two layers of each video are transmitted separately. The base layer is transmitted as one stream over a reliable channel while the enhancement layer is transmitted as a different stream over a less reliable channel. Conceptually, this work implements a rate adaptive multiple description coding. However, it describes only one stream and it does not address the resource management problem arising in multistream transmission scenarios. Reguant *et al.* [11] consider splitting a video stream into two streams and transmitting them over two different broadcast networks. The first stream is transmitted over a DVB-H network at all times while the second stream is transmitted over WiMAX network most of the time. If the user wants to use some other non-video application in parallel, the stream going through WiMAX is degraded to accommodate that application. This ensures a minimum video quality at all times while maintaining the flexibility of using other applications. While this approach has its benefits, it is not very attractive from a deployment point of view since the service provider has to install and manage the infrastructure for two different kinds of networks. Also the solutions described in both [10] and [11] evaluate the performance of video streaming as an application along with other WiMAX applications and do not utilize MBS. In contrast, our approach considers a multimedia-intensive system with extensive use of MBS.

2) Energy Efficient Scheduling: Power-aware scheduling schemes for general WiMAX networks have been proposed in [12]–[15]. For example, Seo *et al.* [12] propose a scheme that

utilizes subscriber information available at the base station. They describe a sleep interval algorithm based on queuing analysis of the packet arrival rate of subscribers. In contrast, Kim *et al.* [13] describe a sleep interval scheme based on the remaining battery life of a mobile subscriber device. Shi *et al.* [14] propose a burst scheduling algorithm for energy minimization on per subscriber basis for unicast data. The algorithm arranges the mobile subscribers in ascending order based on the ratio of the current data arrival rate to the required data rate. If the current rate is significantly higher than the required rate, the mobile subscriber can go to sleep for some interval. After computing the sleep intervals for all mobile subscribers, the bursts are scheduled in a longest interval first manner. After transmission of each burst, the algorithm checks to ensure that the data requirements of all mobile subscribers are being satisfied. The work in [14] is designed for unicast streaming of video and does not consider the multicast/broadcast service. Also the algorithm requires maintaining state information of all mobile subscribers served by a base station.

Liao and Lee [15] suggest a scheduling scheme where the unicast data are clustered around the multicast data bursts for increased energy efficiency. They assume that the burst length and positions for a particular stream is the same in all super-frames. Then they present an enhancement to the longest virtual buffer first scheduling algorithm proposed by Shi *et al.* [14] by clustering the unicast data around the multicast data bursts. Their work evaluates the energy efficiency in a multi-class traffic scenario, whereas our work is focused on the energy efficiency of the video broadcast service.

III. Problem Statement and Hardness

Our work focuses on optimally utilizing the WiMAX Multicast/Broadcast Service to stream multiple scalable videos to mobile receivers. In this section, we state the considered problem and show that it is NP-Complete. We also present the mathematical formulation of the problem. For quick reference, we list all symbols used in the formulation in Table I.

A. Problem Statement

We consider a scenario where a number of scalable video streams are available at a WiMAX base station. Each stream is to be broadcast using MBS to a group of mobile subscribers. At the WiMAX base station, the MBS module allocates a fixed-size data area in the download section of each TDD frame. All video streams are to be allocated only within this MBS data area. As per the mobile WiMAX standard, each MBS data area can transmit a different amount of data depending on the modulation scheme chosen, which is in turn selected based on the wireless channel conditions. For broadcast applications, a common modulation scheme is selected for a group of subscribers. Thus, each MBS area transmits a fixed amount of data, in effect, creating a fixed bandwidth broadcast channel. We consider a scheduling window composed of a number of MBS data areas. Data from the video streams are to be allocated to the MBS areas in the scheduling window. Due to the variable bit rate (VBR) nature of the video streams, the aggregate data rates may exceed the broadcast channel capacity. Hence, in each scheduling window, we need to decide which layers to send for each stream. We assume that the base station has enough buffer space to hold the VBR traffic for one scheduling window. This way, the data rates can be assumed to be constant during a scheduling window, but they vary across scheduling windows. Since the bit rates and the receiver buffer states change in each scheduling window, the allocation has to be computed for every scheduling window. We also assume that all subscribers served by the base station have a fixed amount of buffer which is used to temporarily store the incoming video data before playing it out. Thus, the optimal substream selection problem we need to solve can be stated as follows.

TABLE I
List of Symbols Used in This Paper

Symbol	Description
S	Number of streams
L	Number of layers
q_{sl}	PSNR of substream sl
r_{sl}	Data rate of substream sl
b_{sl}	Number of frame sized blocks of substream sl
n_{sl}	Number of frame bursts for substream sl
t_{sl}^k	Start of burst k of substream sl
w_{sl}^k	Width of burst k of substream sl
τ	Duration of a TDD frame
F	Capacity of MBS data in a TDD frame
P	Number of frames in scheduling window
C	Data capacity of scheduling window
B	Buffer size at the receiver
u_s	Initial buffer level for stream s
E_a	Receiver energy consumption in active state
E_w	Energy consumption for wake up from sleep state
ϵ	Approximation Parameter

1) Problem 1 (Optimal Substream Selection Problem): Select the optimal subset of layers from each scalable stream to broadcast over a WiMAX network such that: 1) the total data transmitted in a scheduling window does not exceed the window capacity, 2) the average quality of all selected substreams is maximized, and 3) the subscriber playout buffer does not suffer from overflow or underflow instances.

B. Problem Hardness

Let us assume that for a given radio modulation scheme, the MBS data area in each frame can accommodate F amount of data and the TDD frame takes τ time to be transmitted. Let the scheduling window consist of P such frames. Then, the maximum amount of data that can be transmitted within the scheduling window is given as $C = PF$. We have S scalable video streams. Each scalable stream s, $1 \leq s \leq S$, has at most L layers. The value of L can be different for each stream. Therefore, for each stream, we have L substreams to choose from, where a substream l includes layer l and all layers below it. Let the data rates and quality values for selecting substream l of stream s be r_{sl} and q_{sl}, respectively. Here r_{11} denotes the data rate of the base layer of the first stream. Thus, we have the problem of choosing the substreams such that the average

quality across the video streams is maximized subject to the following constraints. The first constraint is that the total data to be transmitted must fit into the MBS area in the current scheduling window. The second constraint is that the buffers at the subscribers must not run out of data anytime during the scheduling window, and the third constraint is that the base layer of each stream must be transmitted to guarantee a basic service level agreement.

Theorem 1: The Optimal Substream Selection Problem is NP-Complete.

Proof: First, we consider a relaxed version of the problem with no buffer overflow or underflow constraints. Thus, we are left with the problem of selecting the substreams such that the average quality is maximized. We assume that in each scheduling window at least, all the base layer streams have to be transmitted due to service level agreement. Thus, we further modify the problem by eliminating the base layer constraints, which can be trivially done by reducing the scheduling window capacity by the sum of data rates of all base layers. Therefore, the modified data capacity can be given as $C' = C - \sum_{s \in S} r_{s1}$.

Now we are left with the problem of deciding which substreams to chose from each stream. We show that this problem is equivalent to the NP-Complete 0–1 Multiple Choice Knapsack Problem (0–1-MCKP) [16], which is defined as follows. There are M classes $N_1, \ldots, N_M$ of items to pack in some knapsack of capacity W. Each item (i, j), where $i \in M$, $j \in N_i$, has a profit $p(i, j)$ and a weight $w(i, j)$. The problem is to choose at most one item from each class such that the profit sum is maximized without having the total weight exceed W. We reduce the 0–1-MCKP problem to the Optimal Substream Selection Problem in polynomial time as follows. We make the data rates of choosing a substream represent the item weight and the corresponding quality values represent the profit of choosing an item. We also make the streams represent the multiple choice classes and the scheduling window capacity represents the knapsack capacity. Thus, we have an MCKP instance with S classes, $L - 1$ items per class, and a knapsack capacity of C'. This means that an efficient solution for the simplified Optimal Substream Selection Problem could be employed to efficiently solve the NP-Complete 0–1-MCKP problem. In other words, the substream selection problem is NP-Hard.

In addition, clearly a solution for the simplified Optimal Substream Selection Problem can be verified in polynomial time. Thus, the simplified Optimal Substream Selection Problem is NP-Complete. Consequently, the more general Optimal Substream Selection Problem subject to buffer overflow and underflow constraints is also NP-Complete. ∎

C. Mathematical Formulation

We assume that all subscribers have B amount of buffer available for the video streaming application, and the data rate and quality values for all substreams of each stream are known ahead of the scheduling window. This information can either be obtained as a separate metadata for each stream, or if the scalable video is encoded using H.264/SVC [17] and the base station is media-aware, this information can be obtained directly from the encoded video stream itself using the Supplementary Enhancement Information (SEI) messages. Let the data rate values of substreams be $\{r_{s1}, r_{s2}, \ldots, r_{SL}\}$ and the corresponding quality values be $\{q_{s1}, q_{s2}, \ldots, q_{SL}\}$. Each scheduling window is of duration τP. If substream l of stream s is selected, the amount of data to be transmitted during a scheduling window can be given as $\tau P r_{sl}$. Let binary variables x_{sl} take the value 1 if substream l of stream s is selected for transmission in the current scheduling window and 0 otherwise. For a substream, we define a burst as a consecutive set of MBS data areas allocated to the substream in the scheduling window. For any schedule, let n_{sl} be the number of bursts for substream l of stream s. We denote by variables t^k_{sl} the starting frame number and by variable w^k_{sl} the number of MBS data areas in burst k for substream l of stream s.

The solution of the optimum substream selection problem should generate a list $\langle l, n, \langle t^1_{sl}, w^1_{sl} \rangle, \ldots, \langle t^n_{sl}, w^n_{sl} \rangle \rangle$ for each stream. In the list, l denotes the selected substream, n denotes the number of bursts required for transmitting substream l, and $\langle t^k_{sl}, w^k_{sl} \rangle$ denote the starting point and width of burst k, respectively. For a subscriber receiving channel s, let the buffer level at the beginning of scheduling window be u_s. We need to ensure that all data received during a scheduling window are also consumed in the same window. In other words, $F \sum_{k=1}^{n_{sl}} w^k_{sl} = \tau P r_{sl}$. At the same time, we need to ensure that buffer overflow and underflow do not occur. At the end of each burst, the total data received is given by $F \sum_{i=1}^{k} w^i_{sl}$. During that period, the total data consumed is given by $\tau \left(t^k_{sl} + w^k_{sl}\right) r_{sl}$. Now in order to avoid underflow, the difference of these two terms must be greater than zero for all bursts. Similarly, the overflow conditions can be applied by constraining the difference never to be greater than B. Our objective is to maximize the average video quality over all the streams. We use the peak signal-to-noise ratio (PSNR) values of the streams to denote quality and take an arithmetic average of the PSNRs of the selected streams to denote the average video quality. Let us assume that the data to be transmitted for each substream can be divided into b_{sl} number of F sized data blocks. In other words, $r_{sl} = b_{sl} F$. Consequently, we have the following optimization problem:

$$\text{Maximize} \quad \frac{1}{S} \sum_{s=1}^{S} \sum_{l=1}^{L} x_{sl} q_{sl} \tag{P1}$$

$$\text{such that} \quad \sum_{s=1}^{S} \sum_{l=1}^{L} x_{sl} b_{sl} \leq P \tag{1a}$$

$$\sum_{l=1}^{L} x_{sl} \leq 1 \tag{1b}$$

$$u_s + \sum_{i=1}^{k} w^i_{sl} F - \tau \left(t^k_{sl} + w^k_{sl}\right) r_{sl} \leq B \tag{1c}$$

$$u_s + \sum_{i=1}^{k-1} w^i_{sl} F - \tau \left(t^{k-1}_{sl} + w^{k-1}_{sl}\right) r_{sl} \geq 0 \tag{1d}$$

$$\left[t^k_{sl} \ldots t^k_{sl} + w^k_{sl}\right] \cap \left[t_{\bar{sl}^{\bar{k}}} \ldots t^{\bar{k}}_{\bar{sl}} + w^{\bar{k}}_{\bar{sl}}\right] = \emptyset \tag{1e}$$

$$\sum_{k=1}^{n_{sl}} w^k_s = x_{sl} b_{sl}. \tag{1f}$$

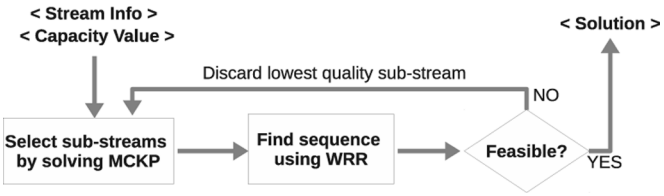

Fig. 2. High-level diagram of the substream scheduling algorithm (SSA).

In the above formulation, the constraint (1a) makes sure that the selected substreams can be transmitted within the broadcast bandwidth. Constraint (1b) ensures that at most one substream is selected for each stream. Constraints (1c) and (1d) represent the buffer overflow and underflow constrains, respectively. Constraint (1e) implies that no two bursts of data blocks should be allocated to the MBS area of the same TDD frame. Here the operator $[\ldots]$ denotes integer interval. This constraint is required because the streams are transmitted over a time-shared, multiple-access wireless channel where only one burst can be transmitted at a time. Constraint (1f) implies that if a layer is selected, then all the data blocks corresponding to the layer must be allocated in the schedule.

IV. Proposed Approximation Algorithm

A. Overview of the Proposed Algorithm

The proposed algorithm is called *Substream Selection Algorithm* and is denoted by SSA. The high level idea of the algorithm is depicted in Fig. 2 and is described as follows. We first find a set of near-optimal substreams given the data capacity of a scheduling window. Then, we allocate them to the MBS areas in the frames of the scheduling window. If no feasible allocation is found, we reduce the problem instance by discarding the substream with lowest quality among all substreams. We solve the optimal substream selection problem again for the reduced set of substreams. This cycle is repeated until either a feasible solution is found, or none of the substreams is selected. Once a solution is found, the frame allocation is done in a modified weighted round robin manner.

As shown in Theorem 1, the problem of selecting optimal scalable substreams is similar to solving the 0–1 Multiple Choice Knapsack Problem. This problem has been studied in the mathematical programming community, and several near-optimal solution schemes exist. The reader is referred to the survey by Lin [18] for a summary of the main results. Dynamic programming is one of the techniques used for designing approximation algorithms for the 0–1 Multiple Choice Knapsack Problem. However, dynamic programming solutions are often memory intensive and may involve large constants. In our algorithm, we derive both an upper bound and lower bound on the value of the solution. These bounds significantly reduce the solution search space.

B. Details of the Proposed Algorithm

Fig. 3 summarizes the pseudo-code of the SSA algorithm. There are three main steps of the algorithm: 1) finding an approximate solution to the substream selection problem, 2) allocating the selected substreams to the MBS data areas of the scheduling window, and 3) validating the schedule to confirm that there are no buffer underflow or overflow conditions. In Fig. 3, the first 11 lines describe the steps of the dynamic programming algorithm for finding the substreams. Lines 12–15 describe the steps for allocating the selected substreams to the MBS areas, and lines 16–19 describe the allocation validation step. Each of these steps are discussed in details in Sections IV-BI–IV-BIII.

Substream Selection Algorithm (SSA)

Input : Substreams, MBS capacity, Frame duration Scheduling window size

Output : Data burst allocation in the MBS area of the current scheduling window

1. For each enhancement layer i across all streams do
2. Compute $\rho_i = r_{sl} - r_{sl-1}$ and $\phi_i = (q_{sl} - q_{sl-1})$
3. Select k largest $\frac{\phi_i}{\rho_i}$ such that $\sum_{i \in k} \rho_i < PF - \sum r_{s1}$
4. Determine lower bound $Q_0 = \sum \phi_i + \sum q_{s1}$
5. Compute scale factor $K = \epsilon Q_0 / S$
6. Scale the quality values such that $q'_{sl} = q_{sl}/K$
7. For $q = 1$ to $2Q_0$ do
8. For $s = 1$ to S do
9. If s is 1, Compute $R(s,q)$ using equation (2a)
10. Else, Compute $R(s,q)$ using equation (2b)
11. Backtrack table $R(s,q)$ to find the substreams s^*
12. Until all streams are allocated do
13. Arrange substreams in ascending order of B_s/τ_s
14. Allocate $\sigma_s = \min B_s/\tau_s$ frames to stream s
15. Update $B_s = B_s + \sigma_s * F - \sigma_s \tau r_s$
16. If no valid allocation found do
17. Find substream $(\hat{l}, \hat{s})$ such that $q_{\hat{s}\hat{l}} = \min_{s \in S, l \in L} \{q_{sl}\}$
18. Discard substream $(\hat{l}, \hat{s})$
19. Go to step 3

Fig. 3. Proposed substream selection algorithm.

1) Approximate Substream Selection: In a naive dynamic programming solution, we construct a table of all possible data rates for the given streams (i.e., $1 \ldots \sum r_{SL}$) and their resulting quality values. For any valid selection of substreams, we define the term *aggregate data rate* as the sum of data rates of the selected substreams. We note that multiple quality values can result for a single aggregate data rate value depending on the composition of the substreams selected. Then, we search for the highest quality entry in the table such that the aggregate data rate is less than the scheduling window capacity. In our proposed algorithm, we first derive bounds on the solution value which will reduce the size of the search space. Then, we construct a dynamic programming table for all quality values within the bounds and find the solution substreams using backtracking.

Bounding the Optimum Solution Value: Solution to the linear programming relaxation of the 0–1 MCKP problem is an upper bound on the optimal solution. Now we derive a lower bound

on the optimal solution as follows. A solution x_c^* to the linear relaxation has the following two properties: 1) x_c^* contains at most two fractional values and 2) when there are two fractional values in x_c^*, they belong to substreams of the same stream. For the proof of these properties, the reader is referred to [19]. Let z_c^* be the value of the objective function corresponding to x_c^*. Let Q_0 be the maximum of 1) the objective function value when both the fractional values are dropped from the solution and 2) maximum of the quality values of the fractional variables. If the optimal solution for the integer problem is Q^*, it is evident that $Q_0 \leq Q^*$. From the properties of x_c^*, it is evident that at most two variables are dropped. Since at most two variables are dropped, z_c^* can be bounded as $z_c^* \leq 2Q_0$. Also, since the solution obtained by the linear relaxation must be greater than or equal to the solution obtained by the integer program, we have an upper bound on the optimum integer solution as $Q_0 \leq Q^* \leq z_c^* \leq 2Q_0$. We note that although the bound is obtained from linear programming theory, we do not require an LP solver to calculate Q_0. Q_0 can be calculated using the median finding algorithms in linear time [20].

Recursive Table Generation: Now that we know the bounds of the optimal solution value, we define a dynamic programming formulation as follows. For all streams $s \in \{1, \ldots, S\}$ and all quality values $q \in, \{0 \ldots, 2Q_0\}$, we define $V(s,q)$ as the set of substreams from streams $1, \ldots, s$ such that no two substreams are selected from the same stream and the total quality of the selected substreams is q. If for a quality value q the *at most one substream per stream* constraint is violated, we set the corresponding sum of weights to infinity. Let $R(s,q)$ denote the sum of data rates selected in $V(s,q)$. We assume that the sum of data rates to produce zero quality is zero, i.e., $R(s,0) = 0$. Also, for the first stream, the data rate values can be computed easily as just the data rate of the substream, or the minimum of the data rates if more than one substream has the same quality, i.e., $R(1,q) = \min_l \{r_{sl}\}$ where $q_{sl} = q$. In mathematical terms, the first stream data rates can be expressed as in (2a). The data rates for the other quality values and other streams can be computed by the recursive definition described in (2b) and the optimum quality can be expressed by (2c):

$$R(1,q) = \begin{cases} \min_l\{r_{sl}\}, & \text{where } l \in L \text{ and } q_{sl} = q, \\ \infty, & \text{otherwise} \end{cases} \tag{2a}$$

$$R(s,q) = \begin{cases} \min\{R(s-1,q), \min_{l \in L}\{r_{sl} \\ \quad +R(s-1,q-q_{sl})\}\}, & \text{when } q_{sl} \leq q, \\ R(s-1,q), & \text{otherwise} \end{cases} \tag{2b}$$

$$Q^* = \max\{q | R(s,q) \leq PF\}. \tag{2c}$$

However, the size of the table can still be very large as it is bounded only by Q_0. Therefore, we select a scaling factor $K = \epsilon Q_0/S$, and scale down the quality values to $q'_{sl} = q_{sl}/K$. This operation considerably reduces the table size while admitting only a small error factor. We bound the quality degradation due to scaling in our mathematical analysis in Section IV-C.

Backtracking: Once we have computed the dynamic programming table, the solution quality value is obtained by a simple scanning of the table as in (2c). The solution substream vectors are found using a backtracking mechanism as follows. While constructing the recursive table, we store the composition of substreams leading to the data rates $R(s,q)$ as a list for each table cell. The solution substream vector is found using the additional information by backtracking from the cell containing the solution quality value.

2) Data Allocation: Once the substreams are selected, it remains to allocate them to the MBS data area such that the subscriber's playback buffers do not overflow or underflow. We use a modified version of the *weighted round robin* algorithm to allocate data to frames. The weighted round robin has been used for scheduling constant bit rate traffic before [21]. However, for a variable bit rate stream, the stream priorities are not static. We derive the priority of a stream based on its buffer level at the subscriber. At the beginning of the scheduling window, for a stream s, let the data rate of the selected substream be r_s and the buffer level be B_s. Then stream s is assigned priority B_s/r_s. A lower value of B_s/r_s denotes higher priority. We also need to allocate the number of frames to a stream in the current round, that is, the length of the burst. The burst length is chosen such that none of the other streams suffers from starvation, nor does it cause overflow or underflow at the receiver buffer. For a stream s, the length of the burst is given by $\min\{B_s/\tau r_s\}$.

3) Buffer State Validation: After the schedule is constructed, we check if any buffer constraint is violated. This can be easily determined by verifying the buffer overflow and underflow constraints described in (1c) and (1d). If the buffer constraints are violated, the current substreams cannot be allocated within the current scheduling window. Hence, we reduce the problem size and re-compute substreams. The problem size is reduced by discarding the substream with minimum quality value among all substreams. This process is repeated until a feasible solution is found or none of the substreams is selected. We note that even though the scheduler is located at the base station, it is aware of the subscriber buffer size and stream data rates. From this information, it can calculate the change in buffer states for a given schedule without any involvement from the subscribers.

C. Correctness and Performance Analysis

We first prove in Lemma 1 that the data rate and quality values of substreams of a scalable stream constitute a non-dominated set. In Lemma 2, we prove the correctness of the recursive formulation described in (2a) and (2b). Using Lemmas 1 and 2, we prove the correctness of SSA.

Lemma 1: Data rates and quality values of substreams extracted from scalable streams constitute non-dominated set.

Proof: Let l and l' be two substreams of a given stream s. Substream l is said to be dominated by substream l' if including l' in the solution always leads to better quality than including substream l. For example, let r_{sl}, r'_{sl} be the data rates and q_{sl}, q'_{sl} be the quality values of substreams l and l'. If $r_{sl} > r'_{sl}$ and $q_{sl} < q'_{sl}$, then l is dominated by l'. Greet *et al.* [22] have shown that, for the H.264 PSNR scalability, when there is sufficient variability in a video, its rate-distortion characterization will be close to a quadratic function which is convex. In our problem, since the streams are variable bit rate videos and layer encoded, the data rate and quality value pair of the layers within a stream can be assumed to form a convex set. ■

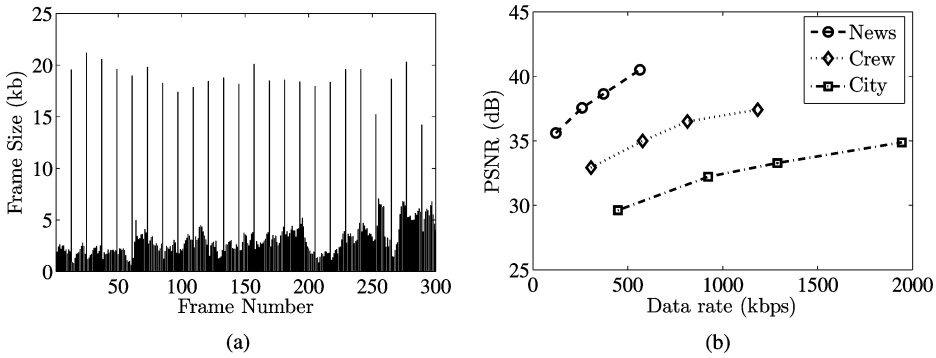

Fig. 4. Rate-distortion characteristics of scalable videos. (a) Foreman sequence. (b) Non-dominating property of scalable videos.

Therefore, Lemma 1 indicates that our problem instances are already in non-dominated form and we can easily solve the linear relaxation of the best quality substream selection problem. Efficient solution to the linear relaxation will help us in efficiently computing the final solution value. We empirically validate the assumption of rate variability and convexity. In Fig. 4(a), we plot the sizes of frames of one of our test streams to show the high data rate variability. In Fig. 4(b), we plot the data rate and PSNR values of three test streams and validate that they indeed form a convex envelope.

Lemma 2: The recurrence relations described in equations (2a)–(2c) produce a near-optimal substream selection solution.

Proof: According to Lemma 1, all instances consist of only non-dominated substreams. Thus, we only need to prove the correctness of the recurrence relation. We can prove the correctness of the recursive expression by induction. The basis step where $s = 1$ is true since it will lead to the selection of the maximum quality substream such that the data rate is less than the scheduling window capacity. Now let us assume that it is also true for the case of $s-1$ streams. For stream s, the expression $R(s-1, q-q_{sl})$ retrieves the weight of the solution and updates it by adding the current data rate. Then all such data rates are compared which can result in quality q and only the minimum is chosen. Since $R(s-1, q)$ is already minimum, this results in $R(s, q)$ also being minimum for every quality value. Since only selections from the set $V(s, q)$ can have non-infinite values, it ensures that only one substream per stream is selected. ■

Theorem 2 (correctness): The Substream Selection Algorithm described in Fig. 3 returns a valid solution for the Substream Selection Problem.

Proof: By Lemma 2, the solution to the dynamic programming formulation selects substreams such that the average quality is close to the optimal and the total data requirement is less than the scheduling window capacity. Therefore, the capacity constraint and the *at most one substream per stream* constraint are satisfied. The round robin algorithm assigns MBS data areas to streams one frame at a time. Thus, it guarantees that no two frame bursts are assigned to the same frame. Finally, the buffer state validation step of the algorithm ensures no buffer overflow or underflow instances occur in the schedule. Hence, the SSA algorithm generates a valid solution for the substream selection problem. ■

Next we analyze the approximation factor and time complexity. We define an approximation algorithm for maximization problems as the following. For every instance I of a maximization problem Π, let $\Pi(I : OPT)$ be the optimal objective function value and let $\Pi(I : ALG)$ be the objective function value obtained by the algorithm ALG. ALG is an approximation algorithm for the problem Π with an approximation factor α if, for any input size, $\Pi(I : OPT)/\Pi(I : ALG) \leq \alpha$. For the problem under discussion, Π corresponds to the substream selection problem.

Theorem 3 (Approximation Factor): The Substream Selection Algorithm described in Fig. 3 is a constant factor approximation algorithm.

Proof: In Section IV-BI, we derived the upper bound and lower bound of the optimal solution in terms of Q_0. For a small nonzero constant ϵ, we selected a scaling factor K as $\epsilon Q_0/S$. Let us consider an instance I of problem Π which has data rate values r_{sl} and quality values q_{sl}. We obtain a scaled down instance I' from I by dividing each quality value q_{sl} by K. Therefore, I' has quality values q'_{sl} which are obtained as

$$q'_{sl} = \lceil q_{sl}/K \rceil .$$

All other aspects of I and I' remain identical. Let $\Pi(I' : ALG)$ be the best solution obtained by any algorithm ALG on the scaled down instance I' and $\Pi(I' : SSA)$ be the solution obtained by our SSA algorithm. Since we rounded up the quality values during the scaling operation, we have

$$\Pi(I : ALG) \leq \Pi(I' : ALG).$$

Also, since our algorithm finds the optimum solution for the scaled down problem, we have

$$\Pi(I' : ALG) \leq \Pi(I' : SSA).$$

From the above two equations, we have

$$\Pi(I : ALG) \leq \Pi(I' : SSA).$$

Now, since each quality value in the solution of $\Pi(I' : SSA)$ is at most K times bigger than the quality values in solution of $\Pi(I : SSA)$, and we can have only S number of such quality values in a valid solution, we have

$$\Pi(I : ALG) \leq \Pi(I' : SSA) \leq \Pi(I : SSA) + SK.$$

Replacing the value of K, we have

$$\Pi(I : ALG) \leq \Pi(I : SSA) + \epsilon Q_0.$$

Now, since Q_0 is a lower bound to our solution, we have

$$\Pi(I : SSA) + \epsilon Q_0 \leq \Pi(I : SSA) + \epsilon \Pi(I : SSA).$$

From the above two equations we have

$$\Pi(I : ALG) \leq (1+\epsilon)\Pi(I : SSA).$$

ALG can be any algorithm including OPT, the optimal algorithm. Therefore, we have

$$\frac{\Pi(I : OPT)}{\Pi(I : SSA)} \leq (1+\epsilon).$$

In other words, the solution obtained by the SSA algorithm is always with a factor of $(1+\epsilon)$ of the optimal algorithm for every instance I of the problem Π. Therefore, the SSA algorithm is a constant factor approximation algorithm with approximation factor $(1+\epsilon)$. ■

Theorem 4 (Time and Space Complexity): The SSA algorithm in Fig. 3 has a time complexity of $O((nS/\epsilon) + Pn \log n)$, where $n = O(\sum L)$ is the total number of substreams, L is the maximum number of substreams within a stream, P is the scheduling window size, and $\epsilon > 0$ is a small constant. The space complexity is $O(SQ^*)$, where S is the number of streams and Q^* is the optimum quality value

Proof: The dynamic programming algorithm computes $S \times 2Q_0$ entries for constructing the $R(s, q)$ table. Computing each entry takes $O(L)$ time. Hence, the table can be completely constructed in $O(L \cdot S \cdot 2Q_0)$ time or $O(nQ^*)$ time. Computation of Q_0 takes $O(n \log n)$ time, leading to a total time complexity of $O(n \log n + nQ^*)$. This is not polynomial in n since the value of Q^* may not be bounded polynomially in n. For a small constant ϵ, we selected a scale factor K and scaled quality value for each substream q'_{sl} according to (3a) and (3b):

$$K = \epsilon Q_0 / S \tag{3a}$$

$$q'_{sl} = \lceil q_{sl}/K \rceil. \tag{3b}$$

Since $Q^* \leq 2Q_0$, we have $Q^*/K \leq 2S/\epsilon$. Thus, the table computation can now be computed in $O(nS/\epsilon)$ time. The round robin allocation takes $O(n \log n)$ time for sorting the buffer levels in each round. The number of rounds depends on the size of the scheduling window P. Hence, the total time complexity is $O((nS/\epsilon) + Pn \log n)$.

Since the optimum solution value Q^* is upper bounded by $2Q_0$, there can be at most $2Q_0$ columns in the dynamic programming table. Also since there is one row for each stream, the number of rows in the table is S. Thus, the table requires $O(SQ^*)$ space for storing the minimum aggregate data rate values pertaining to each quality value. Each cell in the table also needs a constant amount of space to store the backtracking information, but this does not increase the asymptotic space complexity. Hence, the space complexity of the SSA algorithm is $O(SQ^*)$. ■

V. Energy Efficiency for Mobile Subscribers

For mobile receivers, energy savings is one of the major concerns. In this section, we extend our work to provide energy efficient streaming to mobile subscribers. Let the energy consumption for a mobile subscriber while receiving a burst be E_a per TDD frame. Additionally let us assume E_w amount of additional power consumption every time a subscriber has to wake up to receive bursts. We consider the *average energy efficiency (AEE)* metric which is defined as the ratio of energy consumption due to data transfer to the total energy consumption. This AEE metric has been used in previous works such as Shi *et al.* [14]. Our goal is to maximize the AEE metric across all mobile subscribers receiving different streams. To achieve this goal, we add a secondary objective function to the optimization problem given in (P1). This secondary objective function is given by

$$\text{Maximize} \quad \frac{1}{S} \sum_{s=1}^{S} \frac{b_s E_a}{b_s E_a + n_s E_w}. \tag{P2}$$

Clearly, adding the secondary objective function does not reduce the hardness of the problem; it is still NP-Complete. Thus, we propose an approximation algorithm to solve this problem. For the rest of this section, we omit the substream subscripts l from the corresponding terms (e.g., b_s instead of b_{sl}, etc.) for simplicity.

A. Proposed Approximation Algorithm

1) Overview of the Proposed Algorithm: The proposed approximation algorithm is called *Energy Efficient Substream Allocation* and is denoted by EESA. The EESA algorithm is executed after determining the substreams to be transmitted to the subscribers using the SSA algorithm. Thus, instead of sending the selected substreams in a continuous manner, the EESA algorithm will transmit them in bursts in order to save energy for mobile subscribers. The high level idea of the algorithm is as follows. We first assume that the receiver buffer B can be divided into two buffers of size $B/2$ each and the two buffers can be accessed in parallel. This is known as the double-buffering scheme [23]. Since one half of the buffer can be drained while the other half is being filled up in parallel, the scheme always has one buffer for receiving the current burst. Thus, if we stipulate the burst sizes to $B/2$, the buffer overflow problem is resolved. Now if we construct the frames in such a way that the data received in the previous burst is equal to the data consumed during

Energy Efficient Substream Allocation (EESA)

Input : Selected substreams, Initial buffer values, Scheduling window size

Output : Data burst allocation in the MBS area of the current scheduling window

1. For s = 1 to S do
3. Determine $n_s = \lceil 2b_sF/B \rceil$
4. For k = 1 to n_s do
5. Determine x_s^k, y_s^k, and z_s^k using Equations (4a)-(4c)
6. Let $\Lambda = \emptyset$
7. For each decision point do
8. Add a burst from frames t_c to t_n to stream s, where $\mathbf{w}_s^k$
9. has the smallest z_s^k among outstanding bursts, t_c
10. is current time, and t_n is time of the next decision point
11. Let e_s^k be the actual finish time of burst k
12. If $\max\{e_s^k - z_s^k\} \leq 0$ // complete on time
13. Return Λ
14. Return no feasible schedule

Fig. 5. Energy efficient substream allocation algorithm.

the current burst, then we can avoid buffer underflow. Thus, our problem is reduced to finding the number and size of bursts for each stream.

2) Details of Proposed Algorithm: Fig. 5 summarizes the pseudo-code of the EESA algorithm. We calculate the number and size of the bursts for each stream as follows. Since the buffer size is $B/2$, it can accommodate a burst size of at most $\lfloor B/2F \rfloor$. Therefore, we divide the data blocks of each streams into bursts of length $B/2F$. There has to be at least $\lceil 2b_sF/B \rceil$ number of bursts for each stream. We note that the last burst in a scheduling window might have less than $B/2F$ data blocks. We fix the number of bursts n_s to be $\lceil 2b_sF/B \rceil$. For each burst $k \in n_s$, there are a starting frame number, length of burst, and deadline. These are denoted by x_s^k, y_s^k, and z_s^k, respectively. The starting frame number ensures that no data are transmitted before there is sufficient buffer space available in the receiver. Thus, it eliminates the possibility of an overflow. Similarly, the deadline frame numbers ensure that the current burst is transmitted before the receiver buffer runs out of data, eliminating the underflow possibilities. The values of x_s^k, y_s^k, and z_s^k can be derived using the following equations:

$$x_s^k = \begin{cases} \max\left\{0, \frac{u_s}{r_s\tau}\right\}, & \text{for } k = 1 \\ \left\lfloor \frac{(k-1)B}{2r_s\tau} \right\rfloor, & \text{for } 1 < k \leq n_s \end{cases} \tag{4a}$$

$$y_s^k = \begin{cases} \left\lfloor \frac{B}{2F} \right\rfloor, & \text{for } 1 \leq k < n_s \\ b_s - (k-1)\left\lfloor \frac{B}{2F} \right\rfloor, & \text{for } k = n_s \end{cases} \tag{4b}$$

$$z_s^k = \begin{cases} \left\lfloor \frac{kB}{2r_s\tau} \right\rfloor, & \text{for } 1 \leq k < n_s \\ P, & \text{for } k = n_s. \end{cases} \tag{4c}$$

We also define e_s^k as the actual completion frame number for burst k of stream s. When the buffers are initially empty, the starting frame number of the first burst of all streams is zero. However, if the buffers are not empty, the bursts need to wait until the buffers are drained. Each subsequent burst starting point is $\lceil B/2r_s\tau \rceil$ frames away, since this is the time required for the previous burst to be consumed. The end time of each burst can also be derived in a similar way. We define decision points as the time instances at which either a new burst starts, or all frames of a burst has been allocated. These are the two points when we need to decide which burst to send next. At each such decision point, we keep on allocating the burst which has the smallest z_s^k among outstanding bursts.

B. Correctness and Performance Analysis

In the EESA algorithm described in Fig. 5, we use double buffering scheme and fixed length burst to satisfy the buffer overflow and underflow constraints. Also, since the final schedule is constructed by appending the bursts one after the other, there can be no overlap between bursts. Thus, the algorithm produces a valid solution to the energy efficient burst allocation problem.

Theorem 5 (Approximation Factor): The EESA algorithm described in Fig. 5 is a constant factor approximation algorithm with constant given as $\sum_{s=1}^{S}(b_sE_a + 4n_s^*E_w) / (b_sE_a + n_s^*E_w)$, where n_s^* is the optimum number of bursts for stream s.

Proof: To compute the approximation factor, we first determine the number of bursts in the optimal schedule and in the schedule produced by the EESA algorithm. To prevent buffer underflow instances, any optimal burst schedule must have at least $\lceil b_sF/B \rceil$ number of bursts for stream s, that is, $\lceil b_sFB \rceil \leq n_s^*$. On the other hand, because of applying double buffering, the minimum number of bursts required by the EESA algorithm is at least $\lceil 2b_sF/B \rceil$. Now we derive an upper bound on the maximum number of bursts that can be constructed by the EESA algorithm. From the definition of the decision points, each burst derived by (4a)–(4c) can cause at most one interruption in rest of the bursts. This happens when all frames of the current burst are not allocated but the current frame number is the starting frame number of a different burst. Therefore, these interruptions may cause additional $\lceil 2b_sF/B \rceil$ number of bursts for the EESA algorithm. Therefore, the maximum number of bursts created by the EESA algorithm is $2(\lceil 2b_sF/B \rceil)$, in other words, $n_s \leq 4\lceil b_sF/B \rceil$. Comparing this with the value of n_s^*, we have $n_s \leq 4n_s^*$. The approximation factor can be given as the AEE achieved by the optimal algorithm to that achieved by our EESA algorithm. Ignoring the common terms in both expressions, we have the following approximation factor:

$$\frac{\text{AEE(OPT)}}{\text{AEE(EESA)}} = \sum_{s=1}^{S} \frac{b_sE_a + 4n_s^*E_w}{b_sE_a + n_s^*E_w}. \tag{5}$$

We note that since the number of bursts in the optimal solution n_s^* is a constant, the approximation factor is also constant. ■

TABLE II
DATA RATES (KBPS) AND PSNR VALUES (dB) OF THE SCALABLE VIDEOS USED IN EVALUATION

Name	1 Layer		2 Layers		3 Layers		4 Layers	
	r_1	q_1	r_2	q_2	r_3	q_3	r_4	q_4
CREW	306	32.92	578	34.99	814	36.5	1184	37.41
FOOTBALL	442	30.50	827	32.91	1114	33.98	1621	35.55
MOBILE	189	35.76	322	37.87	442	38.93	649	40.36
CITY	448	29.62	923	32.21	1288	33.28	1943	34.88
FOREMAN	170	32.9	407	34.86	589	36.0	890	37.43
BUS	185	33.17	390	35.43	567	36.41	857	37.65
HARBOUR	577	31.8	1025	33.46	1379	34.67	1929	36.25
NEWS	121	35.6	259	37.55	372	38.63	564	40.5
SOCCER	385	29.92	795	32.18	1095	33.13	1651	34.68
ICE	277	32.35	548	34.71	767	35.82	1123	37.32

The allocation algorithm for mobile subscribers, EESA, involves a sorting of the bursts to assign their priorities. Since the number of bursts depends on the length of the scheduling window, sorting takes $O(P \log P)$ time. Therefore, the overall running time of the algorithm EESA is $O(SP \log P)$.

VI. TRACE-DRIVEN EVALUATION

A. Simulation Setup

We have implemented a point-to-multipoint WiMAX multimedia broadcast simulator and evaluated our algorithm in it using actual scalable video traces. For the WiMAX network parameters, we use the 16-QAM modulation scheme with 3/4 convolution turbo coding and 10-MHz channel. Since each TDD frame is 5 ms, for a 1-s scheduling window, we will have to allocate data to 200 TDD frames. Also we assume that within each TDD frame, we have an MBS data area of 50 kb. This gives us a broadcast channel bandwidth of 10 Mbps [24]. At the receiver side, we assume a buffer limit of 512 kb. For generating the video traffic, we use 10 raw (YUV files in 4:2:0 format) video files from the video trace repository of Arizona State University [25]. For each video, we generate a 10-min workload by starting from a random initial frame and then repeating the frame sequences. Then we encode the videos into H.264/SVC format using the JSVM reference software version 9.18 [26]. We encode each stream into four PSNR scalable layers using the medium grain scalability (MGS) feature of the H.264/SVC coding standard [17]. We tune the encoding parameters such that the substreams have average bit rate between 100 kbps and 2.5 Mbps. In Table II, we summarize the information of the data rates and quality values of each layer of the different video files.

B. Simulation Results

1) Video Quality: In our first experiment, we compare the performance of the SSA algorithm versus the optimum algorithm in terms of video quality. We perform this comparison over a period of 100 consecutive scheduling instances. We keep the receiver buffer size fixed at 512 kb, the scheduling window size at 1 s, and vary the number of streams from 10 to 50. Sample results are given in Fig. 6; other results are similar. Fig. 6(a) shows the average quality across all the video streams, whereas Fig. 6(b)–(d) shows individual qualities for the Football, Foreman, and News video sequences, respectively. The figures show that our SSA algorithm produces near-optimal solutions, which are less than 1 dB from the *absolute optimal* solutions computed using the optimization software GLPK [27]. We also observe that the proposed SSA algorithm scales well when the number of streams increase. In another experiment, we keep the number of streams fixed at 20 and vary the scheduling window from 1 to 10 s. As we can see from the results of this experiment in Fig. 7, the solution quality improves as the scheduling window grows. Again, Fig. 7(a) shows the average quality across all the video streams, whereas Fig. 7(b)–(d) shows individual qualities for the Football, Foreman, and News video sequences. These two experiments show that our algorithm produces close to optimal solutions and it scales well, which means that it can support large-scale WiMAX streaming services.

2) Time Efficiency: We evaluate the running time of our algorithms by changing the problem size in two ways. First we keep the scheduling window capacity fixed and increase the number of streams. In a second experiment, we keep the number of streams fixed and increase the scheduling window. We compare the execution times of our algorithms to that of the optimum. For deriving the optimum solution, we first use the GLPK LP solver [27] to determine the substreams that can be scheduled and then sequence the frames within a scheduling window in a weighted round robin manner. For the SSA algorithm, we compute the running time with approximation parameter $\epsilon = 0.01$. For a 1-s scheduling window, we vary the number of streams from 10 to 50 and observe their behavior. The execution times of these algorithms are measured on a computer with 1.6-GHz dual-core processor and 1 GB of memory. The results of the first experiment are shown in Fig. 8(a). As expected, computing the optimum solution using GLPK takes much longer as the number of streams increases. The SSA algorithm runs well within the time window even for large problem instances. For the second experiment, we keep the number of streams fixed at 20 and vary the scheduling window size from 1 to 10 s. From the results of the second experiment, depicted in Fig. 8(b), we can see that the SSA algorithm scales efficiently with increase in window size. For real-time operation, the algorithm needs to compute the solution within the scheduling window duration. From Fig. 8(b), we see that with every 1-s increment in the scheduling window size, the execution time increases by only a few milliseconds. This shows that the SSA algorithm scales well for large problem instances under real-time constraints.

3) Significance of the Approximation Parameter: Next we investigate the effect of the approximation parameter ϵ on the quality of solution and also on the time efficiency of the SSA algorithm. The approximation parameter can be thought of as a knob for tuning the trade-off between solution quality and solution computation time. We first vary the approximation parameter from 0.01 to 0.10 in steps of 0.01 and as seen in Fig. 9(a), the average quality of the received videos degrades as the approximation parameter increases. In Fig. 9(b), we see that when the approximation parameter is increased, the running time of the algorithm decreases. This means that approximate schedules for very large-scale problems can be computed, which can be used

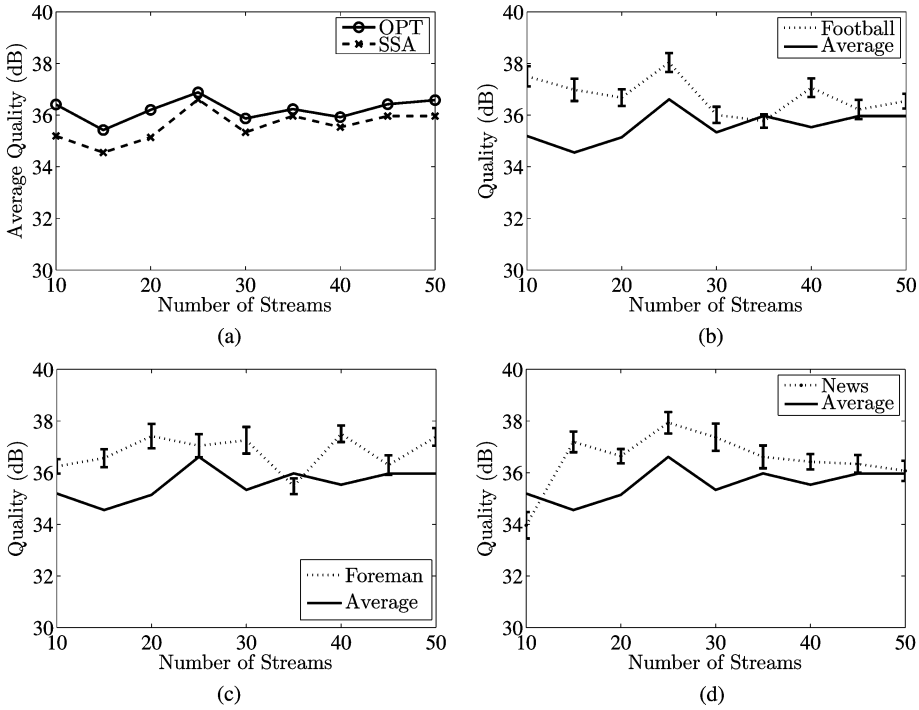

Fig. 6. Near optimality of the solutions obtained by the SSA algorithm. (a) Average. (b) Football. (c) Foreman. (d) News.

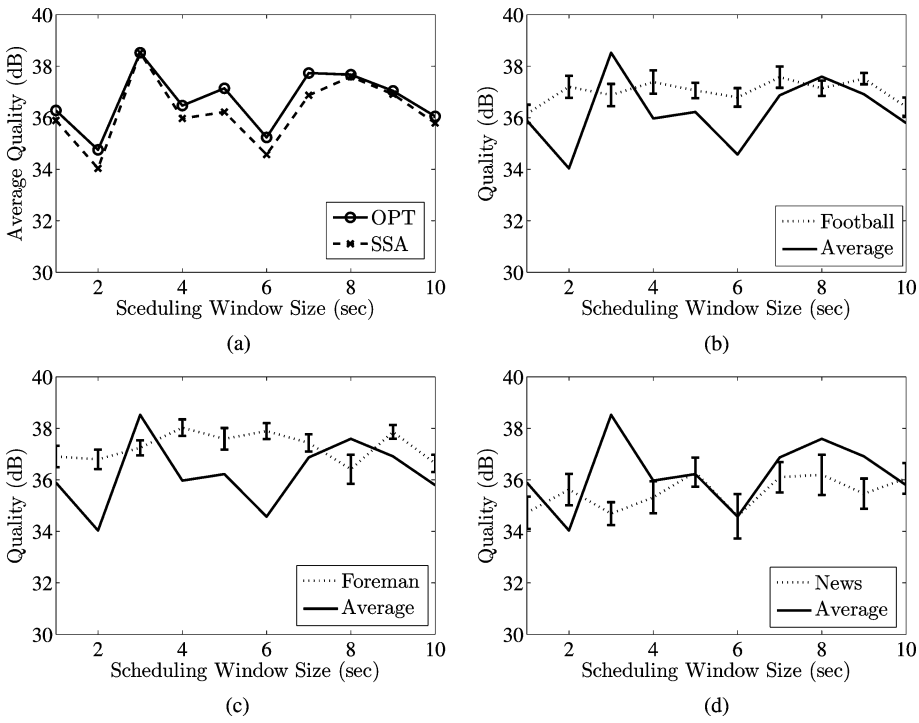

Fig. 7. Effect of scheduling window size on the solutions obtained by the SSA algorithm. (a) Average. (b) Football. (c) Foreman. (d) News.

for preliminary analysis of network deployments. However, in all cases, our algorithm can easily run in real time, even with very small approximation parameters. This is shown in Fig. 9(b) for $\epsilon = 0.01$, and the schedule is computed in less than 0.3 s for a scheduling window of 1 s.

4) Resource Utilization and Buffer Validation: Next, we evaluate the resource utilization of the SSA algorithm in terms of the scheduling window capacity used. Let the total schedulable data capacity of a scheduling window be PF. If $\{\bar{r_1}, \ldots, \bar{r_S}\}$ are the data rates of the chosen substreams, the total data sent in the schedule is $\sum \tau P \bar{r_s}$. The capacity utilization is then given by $\sum \tau P \bar{r_s}/PF$. As seen in Fig. 10, the resource utilization of the SSA algorithm remains close to optimal for different scheduling window capacity sizes.

Next we verify that the buffer conditions are satisfied by the SSA algorithm. That is, we check if the SSA algorithm causes

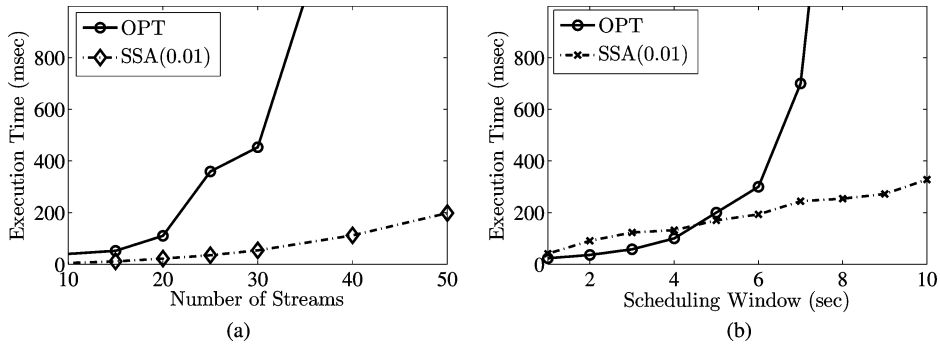

Fig. 8. Running time of the SSA and the optimal algorithms. (a) Fixed window size at 1 s. (b) Fixed number of streams at 20.

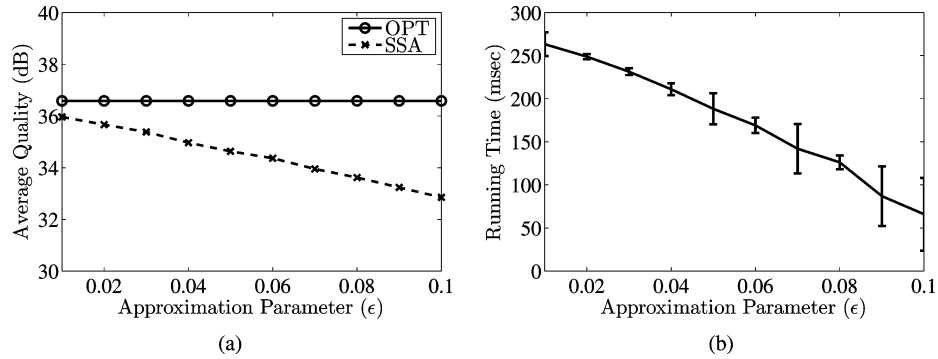

Fig. 9. Effect of the approximation parameter on quality and running time. (a) Fixed number of streams at 50. (b) Fixed number of streams at 50.

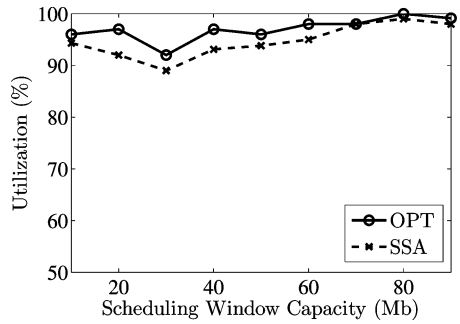

Fig. 10. Performance of the SSA Algorithm in terms of resource utilization.

any buffer overflow or underflow. We found neither overflow nor underflow occurrences for any of the streams. In Fig. 11, we display the buffer level dynamics of subscribers receiving different streams, which show that the buffer level never exceeds 500 kb, that is, there are no overflow instances. It also shows that the buffer level never goes below zero, which means there are no underflow instances. Similar results were obtained for subscribers receiving other video streams.

5) Energy Savings: We evaluate and compare the energy savings resulted from our EESA algorithm to that of the Weighted Round Robin Allocation (WRRA) scheduler. We assume that equal number of mobile stations are receiving each stream. For evaluating the energy saving, we use the Average Energy Efficiency metric. In Fig. 12(a) and (b), we display the comparison between the two algorithms when, respectively, the number of streams and window size are varied. The figures show that the EESA algorithm achieves high values for the AEE metric (close to 1) and remains significantly more efficient than the WRR allocation when the window size increases. The results of another simulation are displayed in Fig. 13 where the impact of receiver buffer size on energy efficiency is measured. We see that the energy efficiency increases when the buffer size is increased. This is directly linked to the fact that the number of switching in a schedule has a high impact on energy savings, because large buffers can absorb larger bursts, which reduces the total number of bursts needed.

VII. Conclusions and Future Work

We presented a framework for multicasting scalable video streams over mobile WiMAX networks. We mathematically analyzed the problem of selecting the optimal substreams of scalable video streams under bandwidth constraints. Solving this problem is important because it enables the network operator to transmit higher quality videos or more number of video streams at the same capacity. We showed that the substream selection problem in presence of bandwidth limitation is NP-Complete. We proposed a novel approximation algorithm for this problem. We proved that our algorithm has a small approximation factor of $(1+\epsilon)$, and it has a time complexity of $O(nS/\epsilon)$, where $n = O(\sum L)$ is the total number of layers, and L is the maximum number of layers in a scalable stream. We implemented and validated our algorithm in a simulation setup and studied the impact of a wide range of parameters using multiple video traces. Our simulation results show that the approximation factor of the proposed algorithm is very close to one for practical scenarios. We also verified that our algorithm can run in real time and that

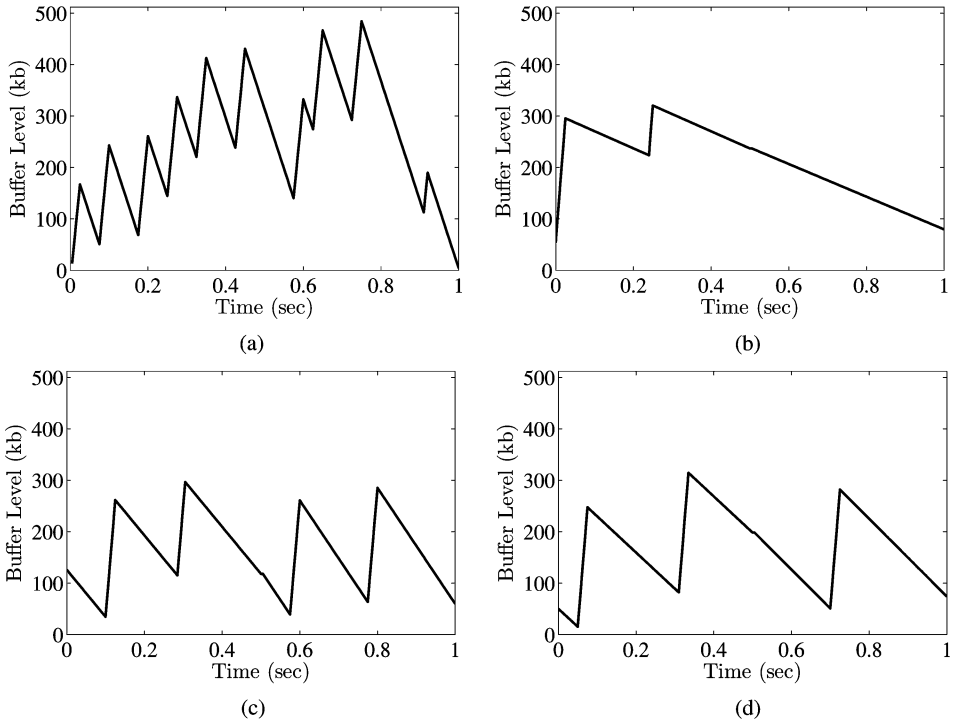

Fig. 11. Performance of the SSA algorithm in terms of receiver buffer dynamics. (a) Football. (b) Foreman. (c) News. (d) Soccer.

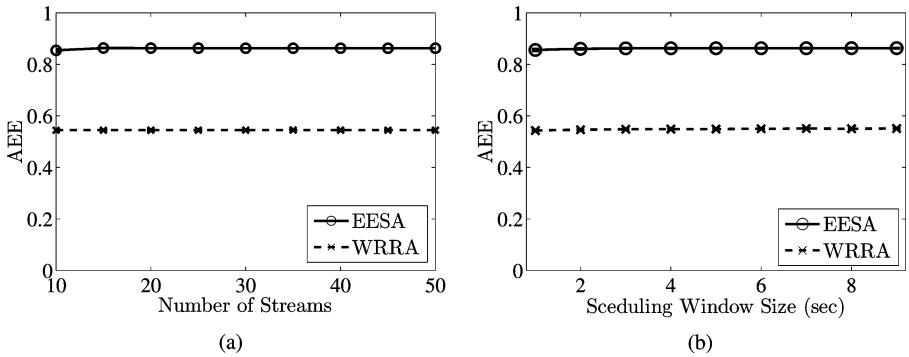

Fig. 12. Energy efficiency of the EESA algorithm.

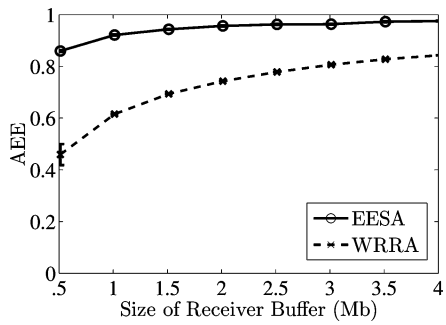

Fig. 13. Effect of buffer size on energy efficiency.

it scales well to large scheduling problems. In addition, we extended our formulation to consider the energy constraints of mobile receivers. We presented an algorithm to transmit the data in bursts in order to conserve energy of mobile receivers. Using simulation, we showed that our algorithm achieves high energy savings.

The work in this paper can be extended in different directions. For example, we are currently extending our algorithm to consider the probability distribution of hardware profiles of active receivers. The algorithm takes into account the diverse parameters like buffer size, display resolution, and energy consumption profiles such that the produced solution not only optimizes the video quality but also enhances the quality of experience for the majority of mobile subscribers.

REFERENCES

[1] Mobile Video Services: A Five-Year Global Market Forecast. [Online]. Available: http://www.pyr.com/store/RPMOBILEVIDEOSERV0906.htm.

[2] Open Mobile Video Coalition website. [Online]. Available: http://www.openmobilevideo.com/resources/omvc-materials/reports/.

[3] *Local and Metropolitan Area Networks Part 16: Air Interface for Broadband Wireless Access Systems Broadband Wireless Metropolitan Area Network.* [Online]. Available: http://standards.ieee.org/getieee802/802.16.html.

[4] Yota Mobile WiMAX Home Page. [Online]. Available: http://www.yota.ru/en/info/main/.

[5] UDCAST WiMAX TV Home Page. [Online]. Available: http://www.udcast.com/products/udcast_wimax_tv_products.htm.
[6] S. Sharangi, R. Krishnamurti, and M. Hefeeda, "Streaming scalable video over WiMAX networks," in *Proc. IEEE Workshop Quality of Service (IWQoS'10)*, Beijing, China, Jun. 2010.
[7] J. Wang, M. Venkatachalam, and Y. Fang, "System architecture and cross-layer optimization of video broadcast over WiMAX," *IEEE J. Select. Areas Commun.*, vol. 25, no. 4, pp. 712–721, May 2007.
[8] R. Cohen, L. Katzir, and R. Rizzi, "On the trade-off between energy and multicast efficiency in 802.16e-like mobile networks," *IEEE Trans. Mobile Comput.*, vol. 7, no. 3, pp. 346–357, Mar. 2008.
[9] P. Hosein, "Broadcasting VBR traffic in a WiMAX network," in *Proc. IEEE VTC'08*, Calgary, AB, Canada, Sep. 2008, pp. 1–5.
[10] H. Juan, H. Huang, C. Huang, and T. Chiang, "Scalable video streaming over mobile WiMAX," in *Proc. Int. Symp. Circuits and Systems*, New Orleans, LA, May 2007, pp. 3463–3466.
[11] V. Reguant, F. Prats, R. de Pozuelo, F. Margalef, and G. Ubiergo, "Delivery of H.264 SVC/MDC streams over WiMAX and DVB-T networks," in *Proc. IEEE Int. Symp. Consumer Electronics (ISCE'08)*, Algarve, Portugal, Apr. 2008, pp. 1–4.
[12] J. Seo, S. Lee, N. Park, H. Lee, and C. Cho, "Performance analysis of sleep mode operation in IEEE 802.16e," in *Proc. IEEE VTC'04*, Los Angeles, CA, Sep. 2004, pp. 1169–1173.
[13] M. Kim, M. Kang, and J. Choi, "Remaining energy-aware power management mechanism in the 802.16e MAC," in *Proc. IEEE Consumer Communications and Networking Conf. (CCNC'08)*, Las Vegas, NV, Jan. 2008, pp. 222–226.
[14] J. Shi, G. Fang, Y. Sun, J. Zhou, Z. Li, and E. Dutkiewicz, "Improving mobile station energy efficiency in IEEE 802.16e WMAN by burst scheduling," in *Proc. IEEE GLOBECOM'06*, San Francisco, CA, Dec. 2006, pp. 1–5.
[15] W. Liao and N. Lee, "An integrated power saving scheduling algorithm in 802.16e wireless metropolitan area networks," in *Proc. Int. Conf. Mobile Technology, Applications & Systems*, Yilan, Taiwan, 2008, pp. 1–8.
[16] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: Freeman, 1979.
[17] T. Wiegand, G. Sullivan, G. Bjøntegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 560–576, Jul. 2003.
[18] E. Lin, "A bibliographical survey on some well-known non-standard knapsack problems," *Inf. Syst. Oper. Res.*, vol. 36, pp. 274–317, Nov. 1998.
[19] A. Sinha and A. Zoltners, "The multiple-choice knapsack problem," *Operations ResearchOper. Res.*, vol. 27, no. 3, pp. 503–515, Jun. 1979.
[20] E. Zemel, "An $O(n)$ algorithm for the linear multiple choice knapsack problem and related problems," *Inf. Process. Lett.*, vol. 18, no. 3, pp. 123–128, Mar. 1984.
[21] P. Hosein and T. Gopal, "Radio resource management for broadcast services in OFDMA-based networks," in *Proc. IEEE Int. Conf. Communications Workshops*, Beijing, China, May 2008, pp. 271–275.
[22] V. Geert, P. David, M. Reisslein, and L. Karam, "Traffic and quality characterization of the H.264/AVC scalable video coding extension," *Adv. MultiMedia*, vol. 2008, no. 2, pp. 1–27, Jan. 2008.
[23] C. Hsu and M. Hefeeda, "Time slicing in mobile TV broadcast networks with arbitrary channel bit rates," in *Proc. IEEE INFOCOM'09*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2231–2239.
[24] D. Gray, Mobile WiMAX Part I—Overview and Performance, WiMAX Forum, 2006. [Online]. Available: http://www.wimax-forum.org/technology/downloads/Mobile WiMAX Part1 Overview and Performance.pdf.
[25] Arizona State University: Video Traces Research Group. [Online]. Available: http://trace.eas.asu.edu.
[26] Fraunhofer HHI: JSVM H.264 AVC/SVC Reference Software. [Online]. Available: http://ip.hhi.de/imagecom G1/savce/downloads/SVC-Reference-Software.htm.
[27] GNU Linear Programming Kit. [Online]. Available: http://www.gnu.org/software/glpk/glpk.html.

Somsubhra Sharangi received the B.Tech. degree from the University of Kalyani, Kalyani, India, in 2003 and the M.Tech. degree from the Indian Institute of Technology Madras, Chennai, India, in 2005. He is pursuing the M.Sc. degree in the School of Computing Science at Simon Fraser University, Surrey, BC, Canada.

His research interests are in the area of multimedia networking and scalable video coding.

Ramesh Krishnamurti received the B.Tech. degree in electrical engineering from the Indian Institute of Technology Madras, Chennai, India, the M.Tech. degree in electrical engineering from the Indian Institute of Technology Bombay, Bombay, India, and the Ph.D. degree in computer and information sciences from the University of Pennsylvania, Philadelphia.

He joined Simon Fraser University, Surrey, BC, Canada, as an Assistant Professor, where he is currently a Professor in the School of Computing Science. His research interests are in combinatorial optimization as well as the design and analysis of algorithms.

Mohamed Hefeeda (S'01–M'04–SM'09) received the M.Sc. and B.Sc. degrees from Mansoura University, Egypt, in 1997 and 1994, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 2004.

He is an Associate Professor in the School of Computing Science, Simon Fraser University, Surrey, BC, Canada, where he leads the Network Systems Lab. His research interests include multimedia networking over wired and wireless networks, peer-to-peer systems, mobile multimedia, and Internet protocols.

Dr. Hefeeda won the Best Paper Award in 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 develop actual systems, such as pCache, svcAuth, pCDN, and mobile TV testbed, and contribute the source code to the research community. The mobile TV testbed software developed by his group won the Best Technical Demo Award in the ACM Multimedia 2008 conference. 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 co-chair the International Conference on Multimedia and Expo (ICME 2011). In addition, he served on many technical program committees of major conferences in his research areas, including ACM Multimedia, ACM Multimedia Systems, and IEEE Conference on Network Protocols (ICNP). He is on the editorial boards of the *ACM Transactions on Multimedia Computing, Communications and Applications* (ACM TOMCCAP), *Journal of Multimedia*, and the *International Journal of Advanced Media and Communication*.