A sleep-mode interleaving algorithm for layered-video multicast services in IEEE 802.16e networks

Chien-Chi Kao a, Shun-Ren Yang b,∗, Hsin-Chen Chen a

a The Department of Computer Science, National Tsing Hua University, Hsinchu 300, Taiwan
b The Department of Computer Science and Institute of Communications Engineering, National Tsing Hua University, Hsinchu 300, Taiwan

ARTICLE INFO
Article history:
Received 10 November 2011
Received in revised form 10 May 2012
Accepted 26 July 2012
Available online 4 August 2012

Keywords:
Bandwidth allocation
IEEE 802.16e
Layered video
Multicast
Sleep mode

ABSTRACT
In this paper we investigate efficient mechanisms to support layered-video multicast services in IEEE 802.16e (Mobile WiMAX) networks. 1 Given the bandwidth eager and energy hungry nature of layered-video multicast services, network systems should employ efficient bandwidth allocation and energy saving mechanisms. We first investigate how the WiMAX energy saving mechanisms significantly degrade the performance of multicast bandwidth allocation mechanisms for layered-video multicast services. Then, we present a theoretical model for illustrating this interaction problem. To the best of our knowledge, this paper offers the first investigation into and the first theoretical model of the interaction problem between multicast bandwidth allocation and WiMAX energy saving mechanisms. To solve the interaction problem, we propose a novel sleep-mode interleaving algorithm beyond the existing mechanisms. The proposed algorithm has full compatibility with the existing multicast bandwidth allocation mechanisms and with IEEE 802.16e standards. By appropriately adjusting one sleep mode parameter defined in the standard, the proposed sleep-mode interleaving algorithm effectively guarantees the bandwidth efficiency of the video multicast mechanisms while mobile users can execute the standard sleep mode operations. Simulation results demonstrate the effectiveness of the proposed algorithm in terms of packet delivery ratio, user satisfaction, energy efficiency and computational complexity.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction
The last few years have seen increasingly rapid advances in the field of broadband wireless communications. As one of the most promising broadband wireless access technologies, IEEE 802.16e [2], i.e., the so-called Mobile WiMAX, has received considerable attention. Mobile WiMAX provides coverage enhancement, bandwidth enhancement, sleep-mode, idle-mode and roaming functions to facilitate mobile access. With all the enhancements and functions, Mobile WiMAX will enable real-time video multicast services in next-generation wireless networks.

Real-time video multicast services, such as mobile IPTV and online gaming, represent fast growing applications. To support such applications while guaranteeing the quality of user experience (QoE) requires both efficient bandwidth allocation and energy saving mechanisms in IEEE 802.16e systems. The reasons for this follow. Mobile users, who subscribe to real-time video multicast services, desire access to high data rate services anytime and anywhere. The high data rate requirements and real time features usually create bandwidth hungry video multicast services. If the limited bandwidth of IEEE 802.16e systems is allocated in an inefficient manner, users would suffer bandwidth starvation, which results in an unsatisfactory QoE. Therefore, employing an efficient bandwidth allocation mechanism becomes a critical issue. On the other hand,
real-time video services signify one of the greatest power consumption applications. When mobile users continuously run real-time video services, they rapidly exhaust their batteries, which also results in an unacceptable QoE. Moreover, this effect diminishes the desire of users to subscribe to real-time video services. Consequently, IEEE 802.16e systems also have the urgent issue to employ an efficient energy saving mechanism.

The literature has presented various approaches to design an efficient bandwidth allocation mechanism for video multicast services. In [3–9], a common notion utilizes the technology of layer encoded video [10] for multicast. This technology allows the splitting of each video stream into a base layer for providing a basic video image and multiple enhancement layers for enhancing video quality. Using this technology, the scheduling approaches can flexibly multicast streaming videos and thereby efficiently allocate the bandwidth. On the other hand, for energy saving, the IEEE 802.16e standard [2] already specifies three classes of sleep mode operations. Among the three classes, the power saving class of type II (PSC II) is recommended for real-time video streaming. According to the specific rules of PSC II, mobile users can enter sleep mode while maintaining real-time video services. Researchers have extensively studied the performance of PSC II. In [11–16], the results validate the operation of PSC II as a promising solution to extend the lifetime for mobile users with real time connections.

Although the literature reports that resource scheduling and sleep mode mechanisms have good individual designs, challenges exist during the concurrent execution of these mechanisms. As an example, consider the Mobile WiMAX network in Fig. 1. The base station (BS) serves two multicast groups \( G_A \) and \( G_B \). The mobile stations (MSs) of \( G_A \) and \( G_B \) subscribe to the video streaming \( V_A \) and \( V_B \), respectively. Assume that the available bandwidth within each scheduling frame (time slot) is 128 Kbps for multicast services. The base layer of each video streaming has the rate of 64 Kbps while each enhancement layer has the rate of 32 Kbps. During the sleep mode operations, suppose that \( G_A \) and \( G_B \) both enter sleep periods at scheduling frames 1, 3 and 5. In this case, during frames 1, 3 and 5, the BS cannot schedule any video streaming to either of the two sleeping groups, therefore wasting the bandwidth of these frames. On the other hand, during frames 2, 4 and 6, both \( G_A \) and \( G_B \) awaken to contend for the available bandwidth. This contention induces \( G_A \) and \( G_B \) to acquire only the base layer (64 Kbps) of \( V_A \) and \( V_B \), respectively. This case points out that the sleep mode operations would significantly affect the performance of the layered video scheduling mechanism.

As a solution to this scenario, we propose interleaving the sleep periods. That is, let \( G_A \) sleep at frames 2, 4 and 6 while \( G_B \) sleeps at frames 1, 3 and 5. In this case, both \( G_A \) and \( G_B \) can acquire the enhancement layers for improving video quality. This example demonstrates that an appropriate sleep-mode interleaving mechanism would effectively guarantee the scheduling performance while the MSs could keep executing the sleep mode operations for energy saving.

To the best of our knowledge, our work represents the first effort to investigate the effect of IEEE 802.16e sleep mode operations on layered video multicast scheduling. Given the unpredictable nature of the sleep mode effect on layered video scheduling, achieving optimal scheduling performance by interleaving the sleep periods would be a tough challenge. To overcome this unpredictability, we first develop a counting method for observing the interaction between the sleep mode operation and the layered video scheduling. Based on the observations, we formally model the sleep-mode interleaving problem as an optimization problem and then propose a novel sleep-mode interleaving algorithm to solve this problem. To reduce the computational complexity, we further present a greedy interleaving algorithm to achieve high performance in polynomial time. Taking practicality into account, the proposed algorithms only adjust one sleep mode parameter without modifying any other configurations of the existing scheduling approaches and of the standard sleep mode operations. We conducted simulations to evaluate the performance of the proposed interleaving algorithms.

The remainder of this paper is organized as follows. Section 2 provides the background information. Sections 3 and 4 present the formulation and solution of the sleep-mode interleaving problem. Section 5 analyzes the computational complexity. Section 6 includes an evaluation of the performance. We discuss the related work in Section 7. Finally, Section 8 concludes the paper.

2. Background

**WiMAX frames:** Symbols in the time domain and subchannels in the frequency domain comprise a WiMAX frame. A grouping of several symbols and one subchannel make up a slot, the minimum resource allocation unit. The slots reserved for multicast and broadcast services form a multicast and broadcast service (MBS) zone.

**Layer encoded videos:** The technology of layer encoding allows the split of each video stream into base layer and multiple enhancement layers. To view a video, users need to receive the base layer. When the users further receive the enhancement layers, the video quality can improve layer by layer. Note that the \( n \)th enhancement layer helps only when users already receive all the \( n-1 \) lower layers.

---

Fig. 1. Layered video multicast example.
**Sleep mode operations:** The IEEE 802.16e standard [2] specifies three power saving classes for sleep mode operations. Among the three classes, PSC II is recommended for the video streaming applications. An MS with real-time video connections can exercise the PSC II operation by starting the sleep windows and listening windows alternately. The standard defines three adjustable parameters: (1) the sleep window length, (2) the listening window length and (3) the start frame number. Before specifying the three parameters, we define a sleep cycle as the combination of one sleep window and one listening window. Within a sleep cycle, the sleep window length equals the number of frames that the MSs doze off and stop sending/receiving data. The listening window length is the number of frames that the MSs wake up to send/receive data. Finally, the start frame number refers to the frame number that the first sleep cycle starts from.

### 3. Problem formulation

#### 3.1. Model and notations

In this paper, we consider network architecture similar to that in Fig. 1. The architecture includes a layer encoded video server, a WiMAX BS and several sleep mode MSs. The video server provides various streaming videos. The videos delivered to the MSs are first buffered at the BS. The BS will make a scheduling decision at the beginning of each WiMAX frame. According to the decision, the BS multicasts the videos to the scheduled groups of MSs. A multicast group is composed of the MSs with the same interest in the video content (e.g., the same TV channel). Let \( G \) be the number of multicast groups in the network and \( N_g \) be the number of MSs in the \( g \)th group. The MSs execute the PSC II operation. Because the MSs within a multicast group should receive the same video contents at the same periods, these MSs must perform the PSC II operation with the same parameters. Accordingly, for the MSs of a multicast group \( g \), \( SW_g \), \( LW_g \) and \( SF_g \) denote the three parameters, respectively, the sleep window length, the listening window length and the start frame number. Fig. 2a shows an example, in which \( SW_g = 2 \), \( LW_g = 3 \) and \( SF_g = 6 \).

#### 3.2. Assumptions

To determine the PSC II parameters needs meeting several quality-of-service (QoS) requirements. Specifically, \( SW_g \) and \( LW_g \) usually relate highly to the QoS measurement (e.g., packet delay and jitter) [11]. To determine the values of \( SW_g \) and \( LW_g \), there already exists several QoS mapping approaches in the higher layer to the MAC layer (see [14] and the references therein). On the other hand, neither these mapping approaches nor the WiMAX standard specifies how to determine the value of \( SF_g \). Changing the parameter, \( SF_g \), would not change the sleep behavior within a sleep cycle and has little effect on the QoS measurement. Therefore, we assume that \( SW_g \) and \( LW_g \) are already determined by one of the existing QoS mapping approaches, and only \( SF_g \) is the adjustable parameter in the remainder of this paper. Note, the same assumption of predetermining \( SWs \) and \( LWs \) (i.e., fixed \( SWs \) and \( LWs \)) was made in the well-known reports [12,13].

To quantify video streaming from the BS to the MSs, we define \( V_{gf} \) as the video data unit that should be transmitted to group \( g \) in frame \( f \). Note that a data unit consists of \( L \) video layers (1 base layer and \( L - 1 \) enhancement layers). For a multicast group in awake (normal) mode, the BS should transmit one data unit to the group once per frame. Note, the same assumption of the layered-video traffic (i.e., \( L \) video layers transmitted to a group once per frame) was also made in the recent reports [6,7].

Based on the assumptions made in the previous reports, we make our assumptions as follows: (1) For a group \( g \) in sleep mode, the BS cannot transmit data to \( g \) during \( g \)'s sleep periods. (2) The unsent data would be first buffered at the BS and then transmitted to \( g \) during \( g \)'s listening periods. (3) The data transmitted to \( g \) will be uniformly distributed among \( g \)'s listening periods. For instance, Fig. 2b shows, the data units \( V_{g,6}, V_{g,7}, V_{g,8}, V_{g,9}, \) and \( V_{g,10} \) uniformly distributed among frames 8, 9, and 10. We denote the amount of the data units actually transmitted to group \( g \) in frame \( f \) as \( D_{gf} \). In Fig. 2, \( D_{g,6} = D_{g,7} = 0 \). \( D_{g,8} = D_{g,9} = 2 \) and \( D_{g,10} = 1 \).

#### 3.3. Constraint and goals

In this paper, the available resources limit the system. Let \( S_f \) be the number of available slots within the MBS zone in frame \( f \). We denote \( S'_{f,lg} \) as the number of slots allocated to group \( g \) for transmitting the \( l \)th layer video in frame \( f \). Then we can express the slot allocation constraint in frame \( f \) as

\[
\sum_{g=1}^{G} \sum_{l=1}^{L} S'_{f,lg} \leq S_f. \tag{1}
\]

![Fig. 2. The sleep mode operation of group g.](image-url)
Under the constraint, we aim to optimize the performance of layered-video multicast scheduling while the MSs activate the sleep mode operations. The performance of both base layer and enhancement layer can reflect the scheduling performance.

The performance of base layer represents a basic guarantee for video subscribers. To prevent any subscriber from starvation, a layered-video multicast scheduling should definitely maintain base-layer delivery ratio at 100% (if possible). Therefore, for the base layer, we aim to multicast the mandatory video streams to all subscribers. Given the limited system resource, the system would fail to transmit base layer streams when the required slots for base layers outnumber the available slots in a certain frame. To avoid the failure of base layer transmission, the required slots of the base layer in each frame should comprise a number less than or equal to the available slots. We can express this objective as

$$\sum_{g=1}^{C} S'_{f,1,g} \leq S_f, \forall f,$$  \hspace{1cm} (2)

where $S'_{f,1,g}$ is the number of slots required by group $g$ to transmit the first layer (i.e., base layer) video stream in frame $f$. To reach this objective, we do not aim to minimize the overall required slots of base layer (i.e., min($\sum_{g=1}^{C} \sum_{f=1}^{F} S'_{f,1,g}$)); instead, we aim to minimize the maximum number of required slots in a certain scheduling frame. In this way, if the maximum number of required slots is minimized and fewer than the number of available slots, the required slots in other scheduling frames would all be ensured that they number fewer than the available slots (i.e., Eq. (2) is satisfied). This is thus a min–max problem, and our goal for the base layer is to minimize the following bottleneck function:

$$\max_{f \in [0, \infty)} \sum_{g=1}^{C} S'_{f,1,g}.$$  \hspace{1cm} (3)

For the enhancement layer, overall user satisfaction can reflect the performance. We define the utility functions, $U_{g,f}$ and $U'_{g,f}$, to represent the satisfaction scores of a group $g$ and an MS $n$ in frame $f$, respectively. Because the satisfaction of a user increases with respect to the video qualities the user gains, the utility $U'_{g,f}$ increases with the number of enhancement layers the MS $n$ receives. In this work, we define $U_{g,f}$ as the total user satisfaction of group $g$, i.e., $U_{g,f} = \sum_{n=1}^{N_g} U'_{g,f}$. From this definition, the more the enhancement layers the group $g$ receives, the more the utility $U_{g,f}$ can be gained. For generalizing our goal of enhancement layer, we aim to maximize the following utility function:

$$\sum_{f=1}^{\infty} \sum_{g=1}^{C} U_{g,f}.$$  \hspace{1cm} (4)

In the remainder of this paper, we refer to the functions of Eq. (3) and Eq. (4) as the bottleneck and utility functions, respectively. Consequently, we intend to minimize the bottleneck and maximize the utility. Note that we will show the goal achievements in Section 6. The achievements of bottleneck minimization and of utility maximization can be verified through base-layer delivery ratio and through user satisfaction, respectively.

3.4. Problem statement

To reach our goals under the system constraint, the problem is how to choose the most appropriate PSC II parameters for the multicast groups. We first observe how the sleep behaviors affect the scheduling performance. Consider Fig. 3 as the example, in which four multicast groups execute the sleep mode operations. We adjust $SF_1$ to 1 and 2 in Fig. 3a and b, respectively. For the convenience of the observation, we design a counting method. We first define the observation range $R$ as the length of a common sleep cycle. Let $SC_g$ be the sleep cycle length of group $g$. In Fig. 3, $R = \text{lcm}(SC_1, SC_2, SC_3, SC_4)$ = lcm(2, 2, 3, 2) = 6. That is, the sleep mode behavior for the four groups will repeat every 6 frames. Then we define a data counting window by a vector $DCW = [DCW_1, DCW_2, \ldots, DCW_6]$ to count the number of data units that should be transmitted within each of the $R$ frames. Note that whichever frame number the data counting window starts from will not affect our observations because the same sleep behavior will repeat every $R$ frames. Let $I$ be the initial frame number of the counting window (e.g., $I = \max(SF_1, SF_2, \ldots, SF_4)$). Then $DCW_1$ is the total number of data units in frame $I$ (i.e., $DCW_1 = \sum_{g=1}^{4} D_{g,1}$). $DCW_2$ is the number of data units in frame $I + 1$ and so on. We use notation $\sigma = \text{stdev}(DCW)$ to represent the standard deviation of the values in $DCW$. When $SF_1 = 1$, $\sigma = 3.31$ (see Fig. 3a). In this case, the large $\sigma$ incurs the bandwidth waste problem in frames 1 and 5, while 9 data units contend for the bandwidth in frame 6. On the other hand, if $SF_1 = 2$, $\sigma = 1.73$ (see Fig. 3b). In this case, the small $\sigma$ leads to no extreme values in the counting window. Consequently, the fewer bandwidth waste and bandwidth contention problems result in the less bottleneck and the more utility (see definitions in Section 3.3). Accordingly, we suppose that the lower the $\sigma$, the less the bottleneck and the more the utility to be gained. To confirm this hypothesis, we formally prove Theorem 1 and Theorem 2. To prove the theorems, we have to make two assumptions and apply two lemmas as follows.

Assumption 1. The multicast groups are all the same size (i.e., $N_1 = N_2 = \cdots = N_6$). We will relax this assumption in Section 3.5.

Assumption 2. Each data unit consumes the same number of slots. This assumption will be relaxed in Section 3.6.

Lemma 1. If $\sigma$ is not minimized, we can find at least two elements $DCW_A$ and $DCW_B$ satisfy the following criteria.

Criterion 1: $DCW_A > DCW_B$.

Criterion 2: By adjusting $SF_g$, $d$ data units can be moved from $DCW_A$ to $DCW_B$.

Criterion 3: After moving the $d$ data units, $\sigma$ decreases.
Lemma 2. For a frame \( f \), the required slots of the base layer are directly proportional to the number of data units, i.e., \( \sum_{g=1}^G S_{1,g} \propto \sum_{g=1}^G D_{g,f} = DCW_{f+1} \).

Theorem 1. The bottleneck \( \max_{f \in \{0, \ldots , R-1\}} \sum_{g=1}^G S_{1,g} \) is minimized if and only if the standard deviation \( \sigma \) is minimized.

Proof. Necessity. Because the sleep behavior will repeat for every \( R \) frames, it is sufficient to show that \( \max_{f \in \{0, \ldots , R-1\}} \sum_{g=1}^G S_{1,g} \) is minimized only if \( \sigma \) is minimized. Suppose \( \max_{f \in \{0, \ldots , R-1\}} \sum_{g=1}^G S_{1,g} \) is minimized and \( \sigma \) is not minimized. Let \( DCW_A \) and \( DCW_B \) be the two elements to satisfy Lemma 1 while \( DCW_A \) is the maximal element of DCW. Adjusting \( SF \) to move \( d \) data units from \( DCW_A \) to \( DCW_B \), will decrease \( \sigma \). After the movement, let \( DCW_{A}^{'} \) be the new maximal element of DCW. Because \( d \) data units are moved from \( DCW_A \) to other elements and the standard deviation of DCW decreases, the value of \( DCW_A^{'} \) must be less than that of \( DCW_A \). By applying Lemma 2, when the maximal number of data units of DCW decreases (from \( DCW_A \) to \( DCW_A^{'} \)), the maximal number of required slots of base layer, \( \max_{g \in \{0, \ldots , R-1\}} \sum_{g=1}^G S_{1,g}^{'} \) will consequently decrease. This results in a contradiction. Therefore, if \( \max_{f \in \{0, \ldots , R-1\}} \sum_{g=1}^G S_{1,g} \) is minimized, \( \sigma \) must be minimized.

Sufficiency. In a similar way, we can prove by contradiction that if \( \sigma \) is minimized, the bottleneck \( \max_{f \in \{0, \ldots , R-1\}} \sum_{g=1}^G S_{1,g} \) is minimized.

Theorem 2. The total utility \( \sum_{f=1}^R \sum_{g=1}^G U_{g,f} \) is maximized if and only if the standard deviation \( \sigma \) is minimized.

Proof. Necessity. Because the sleep behavior will repeat for every \( R \) frames, it is sufficient to show that \( \sum_{f=1}^R \sum_{g=1}^G U_{g,f} \) is maximized only if \( \sigma \) is minimized. Suppose \( \sum_{f=1}^R \sum_{g=1}^G U_{g,f} \) is maximized and \( \sigma \) is not minimized. Let \( DCW_A \) and \( DCW_B \) be the two elements that satisfy Lemma 1 while \( DCW_A \) and \( DCW_B \) locate at frames \( A' \) and \( B' \), respectively. If we adjust \( SF \) to move \( d \) data units from \( A' \) to \( B' \), \( \sigma \) will be decreased. In this case, these \( d \) data units can gain more layers in \( B' \) because the available slots in \( B' \) outnumber those in \( A' \). Also, if these \( d \) data units release their occupied slots in \( A' \), the other data units in \( A' \) can reuse these slots to gain more layers. These gains will increase the total utility. This results in a contradiction. Thereby, if \( \sum_{f=1}^R \sum_{g=1}^G U_{g,f} \) is maximized, \( \sigma \) must be minimized.

Sufficiency. In a similar way, we can prove by contradiction that if \( \sigma \) is minimized, the total utility \( \sum_{f=1}^R \sum_{g=1}^G U_{g,f} \) is maximized.
By applying Theorem 1 and Theorem 2, we can transform both bottleneck minimization and utility maximization problems into the following optimization problem:

\[
\min_{SF} \text{std}(DCW) \quad \text{(5)}
\]

\[
\text{s.t. } 1 \leq SF_g \leq SC_g, \quad \forall g \in G. \quad \text{(6)}
\]

### 3.5. Relaxing Assumption 1: size of multicast groups

In the practical applications, the multicast groups may have diverse sizes. In general, the majority of the existing multicast scheduling approaches [3–9] favor groups with large sizes. This is because the larger group size implies that more users within the group have opportunities to obtain video layers; thus, more users can benefit.

Suppose the scheduling approach favors the large groups and consider the examples in Fig. 4a and b. Suppose group sizes \(N_1 = 10, N_2 = 50, N_3 = 30\) and \(N_4 = 100\). Although \(\sigma\) in Fig. 4a equals that in Fig. 4b, their scheduling performances differ. First, consider the performance of base layer. In Fig. 4a, the bottleneck occurs at frame 6, where the three largest groups (i.e., groups 2, 3 and 4) contend for available bandwidth. In frame 6, the scheduling approach would allocate the base layers to the three groups in decreasing order of group size (i.e., groups 4, 2 and then 3). This case victimizes the 30 users of group 3. On the other hand, in Fig. 4b, by adjusting \(SF_2\) to 2, the bottleneck would occur at frame 3. In this case, only 10 users of group 1 become victims at frame 3. Accordingly, although the \(\sigma_g\) are the same, the users who obtain base layers in Fig. 4b outnumber those in Fig. 4a. Second, consider the performance of enhancement layer. In Fig. 4a, at frames 2 and 4, the scheduling approach would allocate the enhancement layers to group 4 (\(N_4 = 100\)) rather than group 2 (\(N_2 = 50\)). In this case, the 50 users of group 2 become victims. On the other hand, in Fig. 4b, the 50 users of group 2 can receive their enhancement layers at frames 2 and 4. This case victimizes only 10 users of group 1 at frames 5 and 7. Therefore, the total utility gain in Fig. 4b is more than that in Fig. 4a.

By applying the same notion of the counting method that we develop in Section 3.4, we define UCW as the user counting window, which counts the number of competitors within each frame. Let \(\sigma\) be the standard deviation of UCW (i.e., \(\sigma = \text{std}(UCW)\)). The lower \(\sigma\) implies that fewer competitors crowd into the same frame. Consequently, we can infer that the lower \(\sigma\) would result in fewer victims thereby creating more gains of both base layer and enhancement layer (see \(\sigma = 71.41\) in Fig. 4a and \(\sigma = 33.17\) in Fig. 4b). We verify this inference through our simulation results.

### 3.6. Relaxing Assumption 2: slot consumption of data units

In the real world, owing to diverse channel qualities and variable data sizes, data units in a network system usually consumes different numbers of slots. For video multicast services (except for live video services), we suppose that a BS can acquire the information regarding channel qualities and data sizes in advance. Given this information, the BS can accordingly estimate the slot consumption of each data unit (the details will be elaborated on in Section 4.3). We define \(S^g_x\) as the number of slots required by group \(g\) to transmit data units in the \(x\)th frame. Here, for the description purpose, we use the simplified notation \(S^g\) to be the number of slots required by group \(g\) to transmit each data unit (this simplification will be relaxed in Section 4.3). Take Fig. 4c and d as examples, where \(S^g_1 = 10, S^g_2 = 50, S^g_3 = 20\) and \(S^g_4 = 30\). In Fig. 4c and d, although the standard deviations of the amounts of data units are the same (i.e., \(\sigma = 1.73\)), the standard deviations of the amounts of slot consumption are different, which leads to distinct scheduling results. First, we observe the

![Fig. 4. Examples of the user counting window and slot counting window.](image-url)
scheduling results of base layer. In Fig. 4c, the bottleneck happens at frame 6, where two base layers (of group 3) are sacrificed since group 2 consumes too many slots. In contrast, the bottleneck in Fig. 4d happens at frame 3, where all base layers can be served since the slot requirements of groups 1, 3 and 4 are relatively small (compared with group 2). Second, we observe the scheduling results of enhancement layer. In Fig. 4c, we discover that much bandwidth is wasted at frames 5 and 7 since the data units of group 1 consume only a few slots. On the contrary, in Fig. 4d, fewer bandwidth is wasted at frames 2 and 4 since the data units of group 2 consume more slots (than those of group 1). The less waste of slots implies the more slots are utilized for serving both base and enhancement layers; thus, the total utility in Fig. 4d would be higher than that in Fig. 4c.

The above examples reveal that, when the channel and traffic information is available, interleaving by the amount of slot consumption would be more effective than by the amount of data units. This is because the amount of slot consumption can reflect bandwidth allocation more accurately (than the amount of data units). Applying the same notion of DCW and UCW (see Sections 3.4 and 3.5), we further define SCW as the slot counting window, which counts the slot requirements within each frame. Let \( \sigma' \) be the standard deviation of SCW (i.e., \( \sigma' = \text{stdev}(\text{SCW}) \)). The lower \( \sigma' \) implies that the slot requirements are more evenly distributed, which results in fewer bandwidth-contention and bandwidth-waste problems. Accordingly, we can infer that the lower the \( \sigma' \) is, the less the bottleneck and the more the utility are. We also verify this inference through our simulation results.

4. Sleep-mode interleaving algorithm

4.1. Concept of the sleep-mode interleaving algorithm

In this section, we propose the sleep-mode interleaving algorithm aiming to minimize \( \sigma, \sigma' \) and \( \sigma'' \) by properly selecting \( SF_g \) for each multicast group. The interleaving algorithm is executed at the BS when a multicast group arrives/leaves the network or when there is a huge change in the group size. The algorithm determines the appropriate combination of the start frame numbers depending on the information of the pre-configured sleep and listening window lengths. Then, the BS assigns the determined start frame numbers to the MSs. The BS makes the scheduling decision at the beginning of each frame according to the configured sleep-mode parameters. Our interleaving algorithm consists of two phases: (1) minimizing \( \sigma \) or \( \sigma'' \), and (2) minimizing \( \sigma' \). We implement the two phase algorithm by two methods: (1) the brute force method and (2) the greedy method. In both methods, the key step lies in constructing the counting windows, i.e., DCW, UCW and SCW that we develop in Sections 3.4, 3.5 and 3.6.

4.2. Intelligent method for data/user counting window construction

We first elaborate on the construction of DCW. The construction comprises two procedures: (1) predicting the number of data units that should be transmitted to each group \( g \) at each frame within g’s sleep cycle, and (2) counting the total number of data units that should arrive at each frame within the counting window.

To predict the data arrivals for a group \( g \), we define \( D'_{g,x} \) as the number of data units that should be transmitted to \( g \) in the \( x \)th frame of \( g \)’s sleep cycle, where \( 1 \leq x \leq SC_g \). Given \( g \)’s sleep mode parameters, i.e., \( RW_g \) and \( LW_g \), we can simply predict the number of data units \( D'_{g,x} \). Take Fig. 2 as an example. Given \( SW_g = 2 \) and \( LW_g = 3 \), we can predict the data arrivals within \( g \)’s sleep cycle as follows: \( DCW = \{0, 2, 0, 2, 3\} \). The above examples reveal that, when the channel and traffic information is available, interleaving by the amount of slot consumption would be more effective than by the amount of data units. This is because the amount of slot consumption can reflect bandwidth allocation more accurately (than the amount of data units). Applying the same notion of DCW and UCW (see Sections 3.4 and 3.5), we further define SCW as the slot counting window, which counts the slot requirements within each frame. Let \( \sigma'' \) be the standard deviation of SCW (i.e., \( \sigma'' = \text{stdev}(\text{SCW}) \)). The lower \( \sigma'' \) implies that the slot requirements are more evenly distributed, which results in fewer bandwidth-contention and bandwidth-waste problems. Accordingly, we can infer that the lower the \( \sigma'' \) is, the less the bottleneck and the more the utility are. We also verify this inference through our simulation results.

Algorithm 1. DataCounting\((SF, SW, LW)\)

Input:
\( SF = \{SF_1, \ldots, SF_G\}, \quad SW = \{SW_1, \ldots, SW_G\}, \quad LW = \{LW_1, \ldots, LW_G\} \).

Output: \( DCW = \{DCW_1, \ldots, DCW_R\} \).

Procedure:
1: for \( g = 1 \) to \( G \) do
2: \( SCW_g = SW_g + LW_g \)
3: end for
4: \( R = \text{lcm}(SC_1, SC_2, \ldots, SC_G) \)
5: \( I = \text{max}(SF_1, SF_2, \ldots, SF_G) \)
6: for \( f = 1 \) to \( I + R - 1 \) do
7: \( \text{TotalData}_f \leftarrow 0 \)
8: for \( g = 1 \) to \( G \) do
9: \( x \leftarrow [\frac{(f - SF_g) \% SC_g}{R}] + 1 \)
10: \( \text{TotalData}_f \leftarrow \text{TotalData}_f + D'_{g,x} \)
11: end for
12: \( DCW_{f-I+1} \leftarrow \text{TotalData}_f \)
13: end for

To count the number of data units arriving within the DCW, we present an intelligent data counting procedure in Algorithm 1. Algorithm 1 exploits the cyclic property of the sleep mode operation. From line 1 to line 3, we calculate the length of sleep cycle for each group \( g \), i.e., \( SC_g \). At line 4 and line 5, the counting window length and initial frame number are set as \( R = \text{lcm}(SC_1, SC_2, \ldots, SC_G) \) and \( I = \text{max}(SF_1, SF_2, \ldots, SF_G) \) (see the respective definitions and reasons in Section 3.4). From line 6 to line 13, the data units are counted from frame 1 to frame \( I + R - 1 \). For a group \( g \) in frame \( f \), we first map frame \( f \) into the \( x \)th frame within \( g \)’s sleep cycle by using the following equation (line 9):

\[ x = \frac{(f - SF_g) \mod SC_g}{R} + 1. \]  

(7)

Taking Fig. 2 as an example, we map frame 13 into the third frame within the sleep cycle, i.e., \( x = [(13 - 6) \mod 5] + 1 = 3 \). Given the mapping result \( x \) for frame \( f \) and the predicting result \( D'_{g,x} \) for group \( g \), we can derive the total number of data units in frame \( f \), denoted by \( \text{TotalData}_f \), from the following equation (line 10):

\[ \text{TotalData}_f = \sum_{g=1}^{G} D'_{g,x}. \]  

(8)

Finally, since the DCW is located from frame 1 to frame \( I + R - 1 \), the DCW can be constructed as follows (line 12):
$DCW = \{TotalData_0, TotalData_1, \ldots, TotalData_{i-1}\}$ \hfill (9)  
\hfill (10)  
The construction of UCW resembles that of DCW. When replacing the number of data units (that should be transmitted to users) with the number of users (who should receive data units), we can construct the UCW by using all the same procedures as those of DCW.

4.3. Advanced method for slot counting window construction

Compared with the construction of DCW/UCW, it is more difficult to construct an SCW. Different from the construction of DCW/UCW, the construction of SCW includes three parts: (1) predicting the modulation and coding scheme (MCS)$^2$ adopted by the BS for each group; (2) estimating the number of slots required by each group at each frame; (3) counting the total slot requirements at each frame within the counting window.

To predict the MCS adopted by the BS, we propose a systematic procedure in Algorithm 2. First, we define $F_{\text{now}}$ and $CQ_{g,n}^{\text{now}}$ as the current frame and as the current channel quality of user $n$ within group $g$. Second, we define $M$ as the number of all possible MCSs and assume that these MCSs are sorted into increasing order by their bit rates (e.g., from QPSK 1/2 to 64-QAM 3/4). Third, we define an indicator function $I(m, CQ_{g,n}^{\text{now}})$, where $m$ represents one possible MCS among the $M$ candidates. If user $n$ with channel quality $CQ_{g,n}^{\text{now}}$ can receive data by using MCS $m$, then $I(m, CQ_{g,n}^{\text{now}}) = 1$. Otherwise, $I(m, CQ_{g,n}^{\text{now}}) = 0$. Under these definitions, from line 6 to line 12, we first compute $N_{n}^{g}$ as the number of group $g$'s users who can successfully receive data by using MCS $m$. Then, from line 13 to line 19, we predict that $m_{n}^{g}$ would be the MCS adopted by the BS for group $g$. This prediction is based on the fact that the chosen $m_{n}^{g}$ can provide the highest data rate under the constraint of guaranteeing that all users within group $g$ can receive data (see line 15). Note that this kind of MCS is highly likely to be adopted by most of the existing scheduling policies.

Algorithm 2. SlotCounting($SF, SW, LW$)

Input:  
$SF = \{SF_1, \ldots, SF_C\}$, $SW = \{SW_1, \ldots, SW_C\}$,  
$LW = \{LW_1, \ldots, LW_C\}$, $CQ_{g,n}^{\text{now}}$, $D_{g,n}$.

Output:  
$SCW = \{SCW_1, \ldots, SCW_k\}$.

Procedure:  
1: \textbf{for} $g = 1$ to $G$ \textbf{do}  
2: \hspace{0.5cm} $SC_g = SW_g + LW_g$  
3: \textbf{end for}
4: \hspace{0.5cm} $R \leftarrow \text{lcm}(SC_1, SC_2, \ldots, SC_C)$  
5: \hspace{0.5cm} $I \leftarrow F_{\text{now}}$  
6: \hspace{0.5cm} \textbf{for} $m = 1$ to $M$ \textbf{do}  
7: \hspace{1cm} \textbf{for} $g = 1$ to $G$ \textbf{do}  
8: \hspace{1.5cm} \textbf{for} $n = 1$ to $N_g$ \textbf{do}  
9: \hspace{2.0cm} $N_{n}^{g} = N_{n}^{g} + I(m, CQ_{g,n}^{\text{now}})$  
10: \hspace{0.5cm} \textbf{end for}  
11: \hspace{0.5cm} \textbf{end for}  
12: \hspace{0.5cm} \textbf{end for}  
13: \hspace{0.5cm} \textbf{for} $n = 1$ to $G$ \textbf{do}  
14: \hspace{0.5cm} \textbf{for} $m = 1$ to $M$ \textbf{do}  
15: \hspace{0.5cm} \hspace{0.5cm} \textbf{if} $(N_{n}^{m} = N_{n}^{g})$ \textbf{then}  
16: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $m_{n}^{g} = m$  
17: \hspace{0.5cm} \hspace{0.5cm} \textbf{end if}  
18: \hspace{0.5cm} \hspace{0.5cm} \textbf{end for}  
19: \hspace{0.5cm} \hspace{0.5cm} \textbf{end for}  
20: \hspace{0.5cm} \hspace{0.5cm} \textbf{end for}  
21: \hspace{0.5cm} \hspace{0.5cm} $TotalSlot_f = 0$  
22: \hspace{0.5cm} \hspace{0.5cm} \textbf{for} $g = 1$ to $G$ \textbf{do}  
23: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $x = F_{f} + SF_g + 1$  
24: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \textbf{if} $x < I$ \textbf{then}  
25: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $x = x + R$  
26: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \textbf{end if}  
27: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $S_{g,x}^{n} = \left\lfloor \frac{(x - SF_g)}{r_{f,n}} \right\rfloor$  
28: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} \textbf{end if}  
29: \hspace{0.5cm} \hspace{0.5cm} \textbf{end for}  
30: \hspace{0.5cm} \hspace{0.5cm} \textbf{for} $g = 1$ to $G$ \textbf{do}  
31: \hspace{0.5cm} \hspace{0.5cm} \hspace{0.5cm} $TotalSlot_f \leftarrow TotalSlot_f + S_{g,x}^{n}$  
32: \hspace{0.5cm} \hspace{0.5cm} \textbf{end for}  
33: \hspace{0.5cm} \textbf{end for}  
34: \hspace{0.5cm} $SCW_{f-1,G} \leftarrow TotalSlot_f$  
35: \textbf{end for}  

To estimate the slot consumption of group $g$, we first define $r_{f,n}$ as the data rate offered by one slot when the BS adopts MCS $m_{n}^{g}$. Then, let $D_{g,n}^{f}$ be the size of data units that should be transmitted to group $g$ in the $f$th frame. Given $r_{f,n}$ and $D_{g,n}^{f}$, we can thereby derive the slot requirement $S_{g,x}^{n}$ from the following equation (line 27):

$$S_{g,x}^{n} = \left\lfloor \frac{D_{g,n}^{f}}{r_{f,n}} \right\rfloor.$$  \hfill (11)

Finally, from line 20 to line 31, to count the slot requirements within the SCW, we follow the similar procedure as that in Algorithm 1. For group $g$, we first map frame $f$ into frame $x$ by using the following equation (line 23):

$$x = F_{f} + SF_g + 1,$$  \hfill (12)

where $x$ is the relative position after the traffic pattern shifts by $SF_g$ frames. Given the mapping result $x$ and the estimation result $S_{g,x}^{n}$, we can derive the total amount of slot requirements in frame $f$, denoted by $TotalSlot_f$, from the following equation (line 28):

$$TotalSlot_f = \sum_{g=1}^{G} S_{g,x}^{n}.$$  \hfill (13)

Given $TotalSlot_f$ from frame $I$ to frame $I + R - 1$, the SCW can be constructed (line 30):  

$$SCW = \{TotalSlot_1, TotalSlot_{I+1}, \ldots, TotalSlot_{I+R-1}\} = \{SCW_1, SCW_2, \ldots, SCW_k\}.$$  \hfill (14)

4.4. Two-phase sleep-mode interleaving algorithm: brute force method

Algorithm 3 illustrates the brute force method for interleaving the sleep periods of $G$ multicast groups. This
method discovers the minimum $\sigma$ and $\sigma'$ by enumerating all possible combinations of the start frame numbers of the $G$ groups. In the first phase, for each combination, the method constructs the DCW and calculates the corresponding $\sigma$. If $\sigma$ equals the current minimum, the second phase further constructs the UCW and calculates the $\sigma'$. We repeat this two phase procedure until we have examined all the combinations. In this way, we can find the optimal combination. Note that we can construct the DCW and UCW by using the intelligent method proposed in Section 4.2.

For example, suppose we have $G$ multicast groups. The brute force method will construct $SC_1 \times SC_2 \times \cdots \times SC_G$ DCWs with the combinations of different start frame numbers, i.e., \{SF$_1$ = 1 – SC$_1$, SF$_2$ = 1 – SC$_2$, ..., SF$_G$ = 1 – SC$_G$\}. Among all the combinations, if \{SF$_1$ = sf$_1$, SF$_2$ = sf$_2$, ..., SF$_G$ = sf$_G$\} is the optimal combination resulting in the minimum $\sigma$ and $\sigma'$, we choose this combination as the start frame numbers for the $G$ groups.

As discussed in Section 3.6, when the information regarding channel and traffic conditions is available, using SCW would be more effective than using DCW due to the accuracy of measurements. If a system operator prefers interleaving by using SCW, the function DataCounting, the variables DCW and $\sigma$ in Algorithm 3 can be simply replaced by SlotCounting (i.e., the method proposed in Section 4.3), SCW and $\sigma^*$, respectively.

Algorithm 3. Brute force method

\begin{verbatim}
Input: SW = \{SW$_1$, ..., SW$_G$\}, LW = \{LW$_1$, ..., LW$_G$\).
Output: SF = \{SF$_1$, ..., SF$_G$\}.

Procedure:
1: MinStd $\leftarrow \infty$
2: for sf$_1$ = 1 to SC$_1$ do
3: ...$n$
4: for sf$_G$ = 1 to SC$_G$ do
5: DCW $\leftarrow$ DataCounting(\{sf$_1$, ..., sf$_G$\}, SW, LW)
6: $\sigma$ $\leftarrow$ StandardDeviation(DCW)
7: if $\sigma < MinStd$ then
8: MinStd $\leftarrow$ $\sigma$
9: MinUserStd $\leftarrow \infty$
10: $SF$ $\leftarrow$ \{sf$_1$, sf$_2$, ..., sf$_G$\}
11: end if
12: if $\sigma = MinStd$ then
13: UCW $\leftarrow$ UserCounting(\{sf$_1$, ..., sf$_G$\}, SW, LW)
14: $\sigma'$ $\leftarrow$ StandardDeviation(UCW)
15: if $\sigma' < MinUserStd$ then
16: MinUserStd $\leftarrow$ $\sigma'$
17: $SF$ $\leftarrow$ \{sf$_1$, sf$_2$, ..., sf$_G$\}
18: end if
19: end if
20: end for
21: ...
22: end for
\end{verbatim}

4.5. Two-phase sleep-mode interleaving algorithm: greedy method

The brute force method simply examines all possible combinations and thus results in exponential time complexity. To reduce the complexity, we present the greedy method in Algorithm 4. In this method, we choose the locally optimal start frame numbers from group 1 to group $G$. In each greedy step, the method determines a start frame number SF$_g$ by examining the temporary DCWs and UCWs from SC$_G$ combinations of start frame numbers. Among the SC$_G$ combinations, we determine SF$_g$ as the locally optimal number, which results in the minimum $\sigma$ and $\sigma'$. We then repeat this greedy process until we have determined all the $G$ start frame numbers, from SF$_1$ to SF$_G$. Note that we can also construct the temporary DCWs and UCWs by using the intelligent method proposed in Section 4.2.

Algorithm 4. Greedy method

\begin{verbatim}
Input: SW = \{SW$_1$, ..., SW$_G$\}, LW = \{LW$_1$, ..., LW$_G$\).
Output: SF = \{SF$_1$, ..., SF$_G$\}.

Procedure:
1: MinStd $\leftarrow \infty$
2: for $g$ = 1 to $G$ do
3: for sf$_g$ = 1 to SC$_G$ do
4: DCW $\leftarrow$ DataCounting(\{SF$_1$, ..., SF$_{g-1}$, sf$_g$\}, SW, LW)
5: $\sigma$ $\leftarrow$ StandardDeviation(DCW)
6: if $\sigma < MinStd$ then
7: MinStd $\leftarrow$ $\sigma$
8: MinUserStd $\leftarrow \infty$
9: SF$_g$ $\leftarrow$ sf$_g$
10: end if
11: if $\sigma = MinStd$ then
12: UCW $\leftarrow$ UserCounting(\{SF$_1$, ..., SF$_{g-1}$, sf$_g$\}, SW, LW)
13: $\sigma'$ $\leftarrow$ StandardDeviation(UCW)
14: if $\sigma' < MinUserStd$ then
15: MinUserStd $\leftarrow$ $\sigma'$
16: SF$_g$ $\leftarrow$ sf$_g$
17: end if
18: end if
19: end for
20: end for
\end{verbatim}

For example, suppose SF$_1$ is already determined as 1. In this case, the greedy method will construct SC$_2$ temporary DCWs for the first two groups using the combinations of SF$_1$ = 1 and SF$_2$ = 1 to SC$_2$. Among the SC$_2$ combinations, if SF$_1$ = 1 and SF$_2$ = sf$_2$ is the optimal choice resulting in the minimum $\sigma$ and $\sigma'$, SF$_2$ will be determined as sf$_2$. Then, given SF$_1$ = 1 and SF$_2$ = sf$_2$, the greedy method constructs SC$_3$ temporary DCWs for the first three groups using the combinations of SF$_1$ = sf$_1$, SF$_2$ = sf$_2$, and SF$_3$ = 1 to SC$_3$. Therefore, the greedy method will construct SC$G$ temporary DCWs and UCWs for the $G$ multicast groups.
From the $SC_j$ combinations, the algorithm determines the value of $SF_j$, as $s_j$, which leads to the minimum $\sigma$ and $\sigma'$. We repeat this process until we have determined $SF_j$.

Like the brute force method proposed in Section 4.4, to improve the performance of first phase, a system operator can replace the function $DataCounting$, the variables $DCW$ and $\sigma$ in Algorithm 4 with $SlotCounting$ (i.e., the method proposed in Section 4.3), $SCW$ and $\sigma'$, respectively. Note that we will further discuss on the trade-off between scheduling performance and computational complexity in Section 5.

5. Analysis of computational complexity

5.1. Intelligent method for data/user counting window construction

To construct $DCW$ (or $UCW$), we need to count the total number of data units (or users). As shown in Algorithm 1, for each frame, the complexity of counting is $O(G)$. There are $R$ frames within a counting window. Therefore, the total complexity is $O(R \times G)$.

5.2. Advanced method for slot counting window construction

As illustrated in Algorithm 2, during the construction of SCW, we need to predict the MCS adopted by a BS. There are $M$ possible MCSSs and $G$ groups. Each group contains $N_g$ users. Let $N = \max(N_1, N_2, \ldots, N_G)$. The complexity of prediction is $O(M \times G \times N)$. Besides, we need to count the total amount of slot requirements, which takes $O(R \times G)$. Accordingly, the total time complexity is $O(M \times G \times N + R \times G)$. This complexity reveals that constructing an SCW is more time-consuming than constructing a $DCW$/$UCW$.

5.3. Brute force method for sleep-mode interleaving algorithm

In the brute force method, we have to examine $SC_1 \times SC_2 \times \cdots \times SC_C$ combinations of the start frame numbers (see Algorithm 3). For each combination, we need to construct the counting windows. Let the time complexities of constructing a $DCW$/$UCW$ and an $SCW$ be $C_{D/U}$ and $C_S$, respectively. As mentioned in the subsections above, $C_{D/U} = O(R \times G)$ and $C_S = O(M \times G \times N + R \times G)$. Given $SC = \max(SC_1, SC_2, \ldots, SC_C)$, the time complexity of the brute force method is either $O(SC \times C_{D/U})$ or $O(SC \times C_S)$ depending on which counting windows are employed in the interleaving algorithm.

5.4. Greedy method for sleep-mode interleaving algorithm

In the greedy method, we have reduced the computational complexity significantly because we do not have to check all possible combinations. When we choose the start frame number of group $g$, we examine only $SC_G$ combinations. For $G$ groups, the greedy method examines $SC_1 + SC_2 + \cdots + SC_G$ combinations (see Algorithm 4). For each combination, let the time complexities of constructing a $DCW$/$UCW$ and an $SCW$ be $C_{D/U}$ and $C_S$, respectively.

The time complexity of the greedy method is either $O(SC \times G \times C_{D/U})$ or $O(SC \times G \times C_S)$ depending on which construction method is applied.

5.5. Computational cost of re-execution

As the discussions from Section 5.1 to Section 5.4, the time complexity of the proposed interleaving algorithms highly depend on which counting windows are employed. Likewise, the computational cost of re-execution also depends on the chosen counting windows. If employing $DCW$ in the first phase, the proposed algorithms need to be re-executed when a multicast group arrives/leaves the network. If further employing $UCW$ in the second phase, the proposed algorithms need to be re-executed once there is a huge change in the group size. Note that a small change can be neglected in order to conserve the computational resource of a BS. When replacing $DCW$ with $SCW$ in the first phase, the proposed algorithms need to be re-executed once a counting window. This is because the construction of each $SCW$ requires the specific and accurate information regarding channel and traffic conditions. For the proposed algorithms employing $DCW$, $UCW$ and $SCW$, we define $F_D$, $F_U$ and $F_S$ as the respective frequencies of re-execution. In general, $F_D \leq F_U \leq F_S$. The frequency of re-execution (computational cost) is usually directly proportional to the scheduling performance. Therefore, depending on the system operators’ concerns (e.g., computational resource or scheduling performance), the operators can adopt our proposed algorithms while flexibly employing different counting windows.

6. Performance evaluation

6.1. Simulation environment

We evaluate the performance of the proposed sleep-mode interleaving algorithms through simulation experiments conducted by NS2. Table 1 lists the simulation parameters, described as follows. First, we use the realistic system parameters according to the performance study [17] by WiMAX forum. Second, to simulate the PSC II

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel bandwidth</td>
<td>10 MHz</td>
</tr>
<tr>
<td>BS/MS antenna gain</td>
<td>15/–1 dBi</td>
</tr>
<tr>
<td>BS/MS height</td>
<td>32/1.5 m</td>
</tr>
<tr>
<td>BS/MS noise figure</td>
<td>4/7 dB</td>
</tr>
<tr>
<td>Frame duration</td>
<td>5 ms</td>
</tr>
<tr>
<td>Number of slots in MBS zone</td>
<td>80 per frame</td>
</tr>
<tr>
<td>Average data rate of base layer</td>
<td>250 Kbps per layer</td>
</tr>
<tr>
<td>Average data rate of enhancement</td>
<td>250 Kbps per layer</td>
</tr>
<tr>
<td>Number of BSs</td>
<td>1</td>
</tr>
<tr>
<td>Number of multicast groups</td>
<td>5–10</td>
</tr>
<tr>
<td>Number of MSSs</td>
<td>50–500 per group</td>
</tr>
<tr>
<td>Modulation and coding scheme</td>
<td>QPSK 1/2, QPSK 3/4, 16-QAM 1/2, 16-QAM 3/4, 64-QAM 1/2, 64-QAM 2/3, 64-QAM 3/4</td>
</tr>
</tbody>
</table>
operation, we follow the standard of IEEE 802.16e [2]. Third, we choose the opportunistic layered multicasting (OLM) [7] algorithm as the representative bandwidth scheduling mechanism. Fourth, to pre-determine the values of $SW_g$ and $LW_g$, we adopt the QoS mapping approach in [14].

6.2. Input parameters and output measures

In our simulation, we regard $SW_g$ and $LW_g$ as input parameters. To represent the corresponding energy efficiency, we define $r_g$ and $r$ as the energy saving rates for group $g$ and for the whole system, respectively. We can express these rates as

$$r_g = \frac{SW_g}{SC_g},$$

and

$$r = \sum_{g=1}^{G} r_g.$$

To represent the scheduling performance of base layer, we use the base-layer delivery ratio, $B_g$, as an output measure. For the calculation of $B_g$, we first denote $B_{total}$ and $B_{actual}$ as the total number of base layers to transmit and the actual number of base layers transmitted, respectively (see examples in Fig. 3). Then, we can derive $B_g$ as

$$B_g = \frac{B_{actual}}{B_{total}}.$$  

To represent the scheduling performance of enhancement layer, we use the utility function $U_{sf}$ (defined in Section 3.3) as another output measure while setting it as

$$U_{sf} = \sum_{n=1}^{N_s} U_{n,sf},$$

where

$$U_{n,sf} = \sum_{l=1}^{L} I_{n,sf}(l) \times u_l,$$

where $I_{n,sf}(l)$ serves as an indicator function and $u_l$ as the satisfaction score. If MS $n$ receives $l$th layer in frame $f$, $I_{n,sf}(l) = 1$. Otherwise, $I_{n,sf}(l) = 0$. For layer 1 (base layer) to layer 4, $u_l = \{0.4, 0.3, 0.2, 0.1\}$.

6.3. Comparison between brute force method and greedy method

In the first experiment, we simulate our interleaving algorithm by using both the brute force and the greedy methods. In addition to the comparison between these two methods, we simulate three other mechanisms: (1) worst case mechanism, (2) non-sleep mechanism and (3) random mechanism. In the worst case mechanism, we use the brute force method to find the maximum standard deviation. This mechanism stands for the lowest delivery ratio and satisfaction. In the non-sleep mechanism, the MSs always stay in the awake mode without saving energy. In the random mechanism, the MSs implement the PSC II operation while the value of $SF_g$ is randomly chosen for each group. Note that we set the confidence level at 95% with a 3% confidence interval.

In Fig. 5, the number of groups varies from 5 to 10, and the number of users within each group remains fixed at 200. The energy saving rate $r$ is set to be 50% except the non-sleep mechanism ($r = 0\%$). Fig. 5a plots the delivery ratio of base layer data as a function of the number of groups. This figure shows an intuitive result: when the number of groups increases, the delivery ratios decrease due to the contention of the increasing number of multicast groups. In addition, we observe that the non-sleep mechanism achieves the highest delivery ratio because the users in this mechanism remain always awake to receive data. We also observe that both the delivery ratios in the brute force method and in the greedy method are close to the ratio in the non-sleep mechanism, while the delivery ratio in the random mechanism is far from that in the non-sleep mechanism. In addition, the brute force method and greedy method perform better than the worst case by 11.90–19.99% and 11.90–17.40%, respectively. These results indicate that our interleaving algorithm, regardless of method—brute force or greedy method—can effectively guarantee the delivery ratio of base layer data.

Fig. 5b shows another intuitive result: when the number of groups increases, the satisfaction scores increase. This occurs because satisfaction derives from the summation of the scores of users, which increase when the number of groups increases. Also, as we predicted, both the brute-force interleaving algorithm and the greedy interleaving algorithm achieve high user satisfaction. The brute
force method and greedy method outperform the worst case by 4.70–13.50% and 4.30–13.49%, respectively. In contrast, without an interleaving algorithm, the random mechanism would significantly degrade user satisfaction. Note that although the non-sleep mechanism can provide slightly higher satisfaction than our mechanisms, the non-sleep mechanism saves no energy.

Fig. 5c shows that the computational time of the interleaving algorithm increases with the number of groups. Specifically, the execution time of the brute-force method is an exponential growth function of the number of groups, which is clearly impractical. On the contrary, even when there are 10 multicast groups, the greedy method takes less than 1 ms (on the linux system with 2.00 GHz CPU and 4 GB RAM). Due to both the performance and the practicality of the greedy method, we implement our interleaving algorithm using only the greedy method for the following experiments.

6.4. First phase performance: using data counting window

In the second experiment, we simulate our interleaving algorithm with DCW in the first phase. We compare our interleaving algorithm with the random mechanism and the non-sleep mechanism in the scenarios with diverse energy saving rates. In Fig. 6, we have set the number of multicast groups and the number of users within each group to 10 and 200, respectively. The energy saving rate \( r \) varies from 15% to 66%.

Fig. 6a plots the standard deviation of data \( \sigma \) as a function of the energy saving rate \( r \). We can see that our interleaving algorithm achieves much lower \( \sigma \) than the random mechanism. Fig. 6b and c show the results of base layer delivery ratio vs. energy saving rate and total user satisfaction vs. energy saving rate, respectively. The figures show that in all the scenarios, our interleaving algorithm achieves higher delivery ratio and higher user satisfaction than those of the random mechanism. These results demonstrate the effectiveness of the interleaving algorithm in terms of both the base layer and enhancement layer performance.

In Fig. 6, we can also discover that when the \( \sigma \) is lower, both base-layer delivery ratio and user satisfaction are higher. For example, in the case that \( r = 50\% \), the \( \sigma \) reaches one of the lowest points, while both delivery ratio and user satisfaction reach one of the highest points. In contrast, in the case that \( r = 47.8\% \), the \( \sigma \) reaches one of the highest points, while both delivery ratio and user satisfaction reach one of the lowest points. These results confirm our two hypotheses. First, the scheduling performance does not strictly decrease while the energy saving rate increases. This proves the unpredictable nature of the sleep mode effect on scheduling performance. Second, the scheduling performance is affected by \( \sigma \) rather than by \( r \). This result is consistent with our inference drawn in Section 3.4, where we stated that the lower \( \sigma \) would lead to both the less bottleneck and the more utility (verified through the base-layer delivery ratio and through the user satisfaction, respectively).

6.5. First phase performance: using slot counting window

In the third experiment, we design five cases with increasing variances in data rate (see Table 2). Since data units with different rates would consume different numbers of slots, this experiment can examine the effects of slot consumption. In each case, we simulate seven multicast groups, and each group contains 200 users. We set the energy saving rate to 50% (excluding the non-sleep mechanism).

Fig. 7a shows the standard deviation of slots \( \sigma_0 \) under the cases in Table 2. In this figure, we can first see the intuitive result: the standard deviation of slot consumption is directly proportional to the variance of data rates. In all the cases, we can further observe that our interleaving algorithm with SCW always yields lower \( \sigma_0 \) than the random mechanism and the interleaving algorithm with DCW. This result demonstrates that using SCW in the first phase of our interleaving algorithm can effectively reduce \( \sigma_0 \).

Table 2

<table>
<thead>
<tr>
<th>Case</th>
<th>Data rate of each layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10–30 Kbps</td>
</tr>
<tr>
<td>2</td>
<td>10–60 Kbps</td>
</tr>
<tr>
<td>3</td>
<td>10–90 Kbps</td>
</tr>
<tr>
<td>4</td>
<td>10–120 Kbps</td>
</tr>
<tr>
<td>5</td>
<td>10–150 Kbps</td>
</tr>
</tbody>
</table>

Fig. 6. Effects of the energy-saving rate \( r \) on (a) data deviation \( \sigma \), (b) base-layer delivery ratio, and (c) total user satisfaction.
Fig. 7b shows the delivery ratio under the five cases. This figure first shows the intuitive results that when the data rates increase, the delivery ratios decrease (because of the limited bandwidth). In addition, we observe that the proposed interleaving algorithm with SCW achieves higher delivery ratio than not only the random mechanism but also the interleaving algorithm with DCW. This result is consistent with our expectation introduced in Section 3.6, where we infer that lower \( r_0 \) would lead to less bottleneck and thus result in higher delivery ratio.

Fig. 7c shows user satisfaction under the five cases. This figure indicates that when the data rates of data units increase, the satisfaction scores of users decrease. This happens because when the data rates of video layers become higher, the number of layers can be served would become fewer. In this figure, we also observe that our interleaving algorithm with SCW outperforms the random mechanism and the interleaving algorithm with DCW while closely approximating the non-sleep mechanism. This result demonstrates that, by further taking the slot consumption of each data unit into account, our interleaving algorithm with SCW would effectively improve the scheduling performance.

6.6. Second phase performance: using user counting window

In the fourth experiment, we design four cases with increasing variances in group size (see Table 3) to examine the effects of group size. In each case, we simulate seven multicast groups with a diverse number of users within each group. We set the total number of users to 1400 while setting the energy saving rate to 50% (excluding the non-sleep mechanism). The data rate of each layer remains fixed at 250 Kbps so that using whichever counting window in the first phase would not affect the results.

Fig. 8a shows the standard deviation of users \( \sigma^\prime \) under the diverse group sizes with case 1, 2, 3 and 4, as shown in Table 3. In this figure, we can see that when the variance of users becomes large (i.e., case 3 and case 4), our two-phase interleaving algorithm yields lower \( \sigma^\prime \) than the random mechanism and one-phase interleaving algorithm. This result demonstrates that the second phase of our interleaving algorithm can effectively reduce \( \sigma^\prime \).

Fig. 8b shows the base-layer delivery ratio under the four cases. This figure reveals that the one-phase interleaving algorithm and the two-phase interleaving algorithm both achieve a 100% delivery ratio of base layer data, while the random mechanism only reaches about a 90% delivery ratio.

Table 3
The simulated cases in the fourth experiment.

<table>
<thead>
<tr>
<th>Case</th>
<th>( N_1 )</th>
<th>( N_2 )</th>
<th>( N_3 )</th>
<th>( N_4 )</th>
<th>( N_5 )</th>
<th>( N_6 )</th>
<th>( N_7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>2</td>
<td>300</td>
<td>300</td>
<td>200</td>
<td>200</td>
<td>200</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>400</td>
<td>300</td>
<td>200</td>
<td>200</td>
<td>100</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>4</td>
<td>500</td>
<td>400</td>
<td>300</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>50</td>
</tr>
</tbody>
</table>

Fig. 8. Effects of group size on (a) user deviation \( \sigma^\prime \), (b) base-layer delivery ratio, and (c) total user satisfaction.
7. Related work

In this paper, we investigate the interaction problems between the layered-video multicast scheduling mechanisms and sleep mode operations in Mobile WiMAX. We classify the related work into three categories and discuss each as follows.

- **Layered-video multicast scheduling mechanisms in mobile WiMAX:** The literature has proposed numerous WiMAX scheduling mechanisms [3–9]. For supporting the layered-video multicast services, the authors in [3] first presented a two-level scheduling mechanism, which allocated bandwidth to the base layer videos in the first level and then to the enhancement layer videos in the second level. To improve the QoE, [4,5] presented two utility-oriented scheduling mechanisms, where the proposed mechanisms allocated bandwidth with the aim of maximizing the different utility functions they defined. In [6], Huang et al. argued for an enhanced system throughput by using opportunistic multicasting. Based on the results in [6], the same authors in [7] developed a complete opportunistic scheduling mechanism, OLM (Opportunistic Layered Multicasting), for layered-video multicast in Mobile WiMAX. On the other hand, based on the utility-oriented scheme in [4], Kuo et al. in [8] also presented their latest work on designing an adaptive resource allocation scheme to maximize all users’ satisfaction. For future WiMAX networks, Sheu et al. in [9] have started to propose novel scheduling mechanisms to support layered-video multicast in WiMAX relay networks. Among these scheduling mechanisms, because OLM in [7] represents one of the most solid scheduling mechanisms for the current IEEE 802.16e systems, we chose OLM as the representative mechanism in our simulations.

- **Sleep mode operations in mobile WiMAX:** To investigate the energy saving mechanisms in Mobile WiMAX, many studies [18,11–13,19–22] were reported. In [18] and the references therein, several techniques were developed to examine the performance of the power saving class of type I (PSC I). In [11], Kong and Tsang first presented a performance comparison between the different operations, i.e., PSC I for non-real-time services and PSC II for real-time services. In [12], Chen et al. presented the first work to focus on PSC II, and they proposed an algorithm to maximize the energy efficiency of PSC II. In [13], to improve practicality, the same authors introduced two mathematical methods to reduce the computational complexity of their algorithm in [12]. To examine the practicality of WiMAX sleep mode operations, Cicconetti et al. in [19] conducted extensive packet-level simulations in the context of the IEEE 802.16e. For the future IEEE 802.16 m systems, several prior studies [20–22] have been published and conducted different performance modeling of IEEE 802.16 m sleep mode operations, while almost all (if not all) of them claimed that IEEE 802.16 m will provide backward compatibility with IEEE 802.16e. Although these reports demonstrated the energy efficiency of WiMAX sleep mode operations, Chen et al. in [13] and Cicconetti et al. in [19] noted the need for further research on many issues when taking bandwidth scheduling and QoS into account.

- **Interaction problems between bandwidth scheduling and sleep mode operations in mobile WiMAX:** In comparison with the large number of studies on WiMAX scheduling mechanisms or on WiMAX power saving mechanisms, a small yet increasing number of reports [24–26,14–16,1] looked at the interaction problems between these two highly related mechanisms. In [24], we indicated that with proper scheduling priority, the scheduling mechanisms can provide high energy efficiency to MSs. Similarly, the authors of [25] proposed a priority-based scheduling mechanism to strike a balance between energy efficiency and scheduling performance. In [26], Huang et al. suggested the sleep mode MSs share the same sleep cycle so that the MSs can be simply scheduled in an energy efficient manner. In [14], Chen et al. argued that the common sleep cycle in [26] restricts MSs’ behaviors, and proposed a packet-filling algorithm to properly determine the length of the sleep cycle for each MS according to the MS’s delay bound. Based on the delay-awareness algorithm in [14], Chen et al. in [15] further took packet inter-arrival time into account and accordingly incorporated a new policy into their algorithm to pursue higher energy efficiency and resource utilization. Then, considering multiple connections within a single MS, Tseng et al. in [16] explored the concept of [15] to redesign a method for each sleep-mode MS to manage its own connections. Although some offered solutions [24–26,14–16] to the interaction problems, all the solutions tackled the single layer and unicasting traffic only. For layered-video multicast services, we published the first report in [1], where we introduced the concept of the sleep-mode interleaving algorithm for the first time.

In this paper, we extend our previous work [1] in the following ways: (1) Instead of purely maximizing the overall user satisfaction, the goal of this paper is not only to...
maximize the utility but also to minimize the bottleneck. (2) In this paper, we have added the theoretical analysis of the bottleneck problem. (3) In addition to relaxing the assumption on group size, we have further relaxed the assumption on slot consumption. (4) The construction of the counting windows (i.e., DCW and UCW) was not specifically considered in [1]. In this paper, we have proposed an intelligent method to construct these windows in an efficient way. (5) Moreover, to improve the schemes’ performances in [1], we have introduced a novel concept of SCW and accordingly proposed an advanced method to construct SCWs. (6) Taking the limited computational resources into account, we have added a new section for discussing on the computational complexities of the proposed schemes. (7) In addition to the theoretical analysis, we have also conducted more simulation experiments to evaluate the performance of the proposed schemes. (8) Finally, Section 7 of this paper has provided the latest information regarding the scheduling mechanisms and the sleep-mode operations in Mobile WiMAX systems, which gives a better view of the work.

8. Conclusion

To the best of our knowledge, our work is the first attempt to tackle the interaction problem between layered video scheduling and sleep mode operation. Although several (and an increasing number of) approaches were proposed in the literature to determine the sleep window length and the listening window length, only a few reports looked at how to properly determine the start frame number. In this work, we first illustrated the following: (1) without proper start frame numbers, the sleep mode effect would significantly degrade the scheduling performance; and (2) the optimization problem of how to determine a proper set of start frame numbers is non-trivial. Then, we solved the problem in a systematic way and proposed the sleep-mode interleaving algorithm. The proposed algorithm is fully compatible with the existing QoS mapping approaches and with the state-of-the-art scheduling mechanisms.

This paper presented the advanced model and solution of the interaction problem between the layered-video multicast scheduling mechanisms and sleep mode operations in IEEE 802.16e networks. Simulation results demonstrated the following: (1) without an interleaving algorithm, the sleep mode effect significantly degrades the scheduling performance; (2) our sleep-mode interleaving algorithm achieves both the high delivery ratios and high user satisfaction in various scenarios; (3) the greedy method for solving the interleaving problem effectively guarantees scheduling performance while significantly reducing the computational complexity of the brute force solution; (4) when channel and traffic conditions are accessible, estimating slot consumption in the first phase of the interleaving algorithm can improve scheduling performance; and (5) the second phase of sleep-mode interleaving algorithm further improves user satisfaction when the multicast group sizes are non-uniformly distributed.

In this paper, we evaluate our schemes’ performances through an experimental lower bound, i.e., the worst case mechanism. As our future work, we will try to examine the feasibility of developing mathematical models for analyzing the performances of our proposed schemes.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments, which have significantly enhanced the quality of this paper. This work was supported in part by the National Science Council of Taiwan under Contract 98-2221-E-007-058-MY2, Contract 98-2221-E-009-059-MY2, and Contract 100-2219-E-007-002-.

References


Chien-Chi Kao received the B.S. degree in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 2006 and the M.S. degree in communications engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2008. He is currently working toward the Ph.D. degree with the Department of Computer Science, National Tsing Hua University. His current research interests include wireless communications and mobile computing. He is a student member of the IEEE and an honorary member of the Phi Tau Phi Society.

Shun-Ren Yang received the B.S. and M.Sc. degrees in computer science and information engineering and the Ph.D. degree from National Chiao Tung University, Hsinchu, Taiwan, in 1998, 1999, and 2004, respectively. From April 1, 2004 to July 31, 2004, he was a Research Assistant with the Department of Information Engineering, Chinese University of Hong Kong, Shatin, Hong Kong. Since August 2004, he has been with the Department of Computer Science and the Institute of Communications Engineering, National Tsing Hua University, Hsinchu, where he is currently an Associate Professor. His current research interests include the design and analysis of personal communications service networks, computer telephony integration, mobile computing, and performance modeling. He is a member of the IEEE Computer and IEEE Communications Societies.

Hsin-Chen Chen received the B.S. degree in computer science from National Tsing Hua University, Hsinchu, Taiwan, in 2010. He is currently working toward the M.S. degree with the Department of Computer Science, National Tsing Hua University. His current research interests include wireless communications and mobile computing.