A Cross-Layer Framework for Overhead Reduction, Traffic Scheduling, and Burst Allocation in IEEE 802.16 OFDMA Networks

Jia-Ming Liang, *Student Member, IEEE*, Jen-Jee Chen, *Member, IEEE*, You-Chiun Wang, *Member, IEEE*, and Yu-Chee Tseng, *Senior Member, IEEE*

***Abstract*—IEEE 802.16 orthogonal frequency-division multiple access (OFDMA) downlink subframes have a special 2-D channel-time structure. Allocation resources from such a 2-D structure incur extra control overheads that hurt network performance. Existing solutions try to improve network performance by designing either the scheduler in the medium access control layer or the burst allocator in the physical layer, but the efficiency of overhead reduction is limited. In this paper, we point out the necessity of "codesigning" both the scheduler and the burst allocator to efficiently reduce overheads and improve network performance. Under the partial-usage-of-subcarriers model, we propose a cross-layer framework that covers overhead reduction, real-time and non-real-time traffic scheduling, and burst allocation. The framework includes a two-tier priority-based scheduler and a bucket-based burst allocator, which is more complete and efficient than prior studies. Both the scheduler and the burst allocator are tightly coupled together to solve the problem of arranging resources to data traffic. Given available space and bucket design from the burst allocator, the scheduler can well utilize the frame resource, reduce real-time traffic delays, and maintain fairness. On the other hand, with priority knowledge and resource assignment from the scheduler, the burst allocator can efficiently arrange downlink bursts to satisfy traffic requirements with low complexity. Through analysis, the cross-layer framework is validated to give an upper bound to overheads and achieve high network performance. Extensive simulation results verify that the cross-layer framework significantly increases network throughput, maintains long-term fairness, alleviates real-time traffic delays, and enhances frame utilization.**

***Index Terms*—Burst allocation, cross-layer design, fair scheduling, IEEE 802.16, Worldwide Interoperability for Microwave Access orthogonal frequency-division multiple access (WiMAX OFDMA).**

Manuscript received March 15, 2010; revised January 20, 2011; accepted February 25, 2011. Date of publication March 10, 2011; date of current version May 16, 2011. The work of Y.-C. Tseng was supported in part by the Aiming for the Top University and Elite Research Center Development Plan, by the National Science Council under Grant 97-3114-E-009-001, Grant 97-2221-E-009-142-MY3, Grant 98-2219-E-009-019, Grant 98-2219-E-009-005, and Grant 99-2218-E-009-005, by the Industrial Technology Research Institute, Taiwan, by the Institute for Information Industry, Taiwan, by D-Link, and by Intel Corporation. The review of this paper was coordinated by Dr. N. Ansari.

J.-M. Liang, Y.-C. Wang, and Y.-C. Tseng are with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu 30010, Taiwan (e-mail: jmliang@cs.nctu.edu.tw; wangyc@cs.nctu.edu.tw; yctseng@cs.nctu.edu.tw).

J.-J. Chen is with the Department of Electrical Engineering, National University of Tainan, Tainan 70005, Taiwan (e-mail: james.jjchen@ieee.org).

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

Digital Object Identifier 10.1109/TVT.2011.2125808

I. Introduction

The IEEE 802.16 standard [1] has been developed for wide-range broadband wireless access. The physical (PHY) layer employs the orthogonal frequency-division multiple access (OFDMA) technique, where a *base station* (BS) can simultaneously communicate with multiple *mobile subscriber stations* (MSSs) through a set of orthogonal subchannels. The standard supports the frequency-division duplex (FDD) and the time-division duplex (TDD) modes. This paper aims at the TDD mode. Under the TDD mode, the following two types of subcarrier grouping models are specified: 1) *adaptive modulation and coding* (AMC) and 2) *partial usage of subcarriers* (PUSC). AMC adopts a contiguous permutation strategy, which chooses adjacent subcarriers to constitute each subchannel and leverages channel diversity by the high correlation in channel gains. However, each MSS needs to report its channel quality on every subchannel to the BS. On the other hand, PUSC adopts a distributed permutation strategy, which randomly selects subcarriers from the entire frequency spectrum to constitute each subchannel. Thus, the subchannels could be more resistant to interference, and each MSS can report only the average channel quality to the BS. Because PUSC is more interference resistant and mandatory in the standard, this paper adopts the PUSC model. In this case, there is no issue of subchannel diversity (i.e., the qualities of all subchannel are similar), because the BS calculates the average quality for each subchannel based on MSSs' reports [2], [3].

The BS manages network resources for MSSs' data traffic, which is classified into *real-time traffic* [e.g., unsolicited grant service (UGS), real-time polling service (rtPS), and extended real-time polling service (ertPS)] and *non-real-time traffic* [e.g., non-real-time polling service (nrtPS) and best effort (BE)]. These network resources are represented by *frames*. Each frame consists of a *downlink subframe* and an *uplink subframe*. Each downlink subframe is a 2-D array over channel and time domains, as shown in Fig. 1. The resource unit that the BS allocates to MSSs is called a *burst*. Each burst is a 2-D subarray and needs to be specified by a *downlink map information element* (*DL-MAP_IE* or simply *IE*) in the downlink map (DL-MAP) field. These IEs are encoded by the robust quaternary phase-shift keying (QPSK) 1/2 modulation and coding scheme (MCS) for reliability. Because the IEs occupy frame space and do not carry MSSs' data, they are considered *control overheads*. Explicitly, how we can efficiently reduce IE overheads will

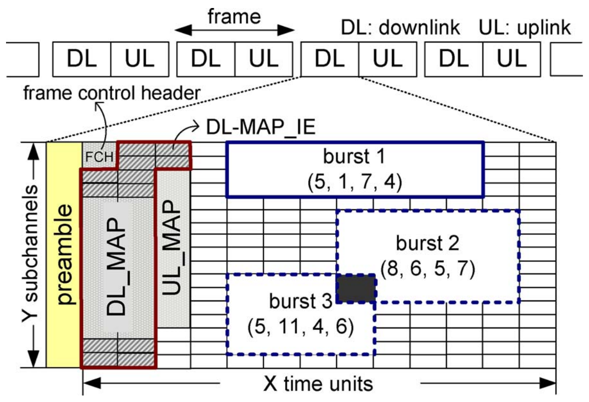


Fig. 1. Structure of an IEEE 802.16 OFDMA downlink subframe under the TDD mode.

significantly affect network performance, because it determines frame utilization. To manage resources to all data traffic, the standard defines a *scheduler* in the medium access control (MAC) layer and a *burst allocator* in the PHY layer. However, their designs are left as open issues to implementers.

This paper aims at *codesigning* both the scheduler and the burst allocator to improve network performance, which covers overhead reduction, real-time and non-real-time traffic scheduling, and burst allocation. The design of the scheduler should consider the following three issues.

- The scheduler should improve network throughput while maintaining long-term fairness. Because the BS may send data to MSSs using different transmission rates (due to network situations), the scheduler will prefer MSSs that use higher transmission rates but should avoid starving MSSs that use lower transmission rates.
- The scheduler should satisfy the delay constraints of real-time traffic to avoid high packet-dropping ratios. However, it should also meet the requirements of non-real-time traffic.
- To well utilize the limited frame space, the scheduler has to reduce IE overheads when assigning resources to MSSs' data traffic. This condition requires the knowledge of available frame space and burst arrangement design from the burst allocator.

On the other hand, the design of the burst allocator should address the following three issues.

- The burst allocator should arrange IEs and downlink bursts for the MSSs' resource requests from the scheduler in the OFDMA channel-time structure to well utilize the frame space and reduce the control overhead. Under the PUSC model, because all subchannels are equally adequate for all MSSs, the problem of arranging IEs and downlink bursts will become a 2-D mapping problem, which is NP-complete [4]. To simplify the burst arrangement problem, an advance planning for the MSSs' resource requests in the scheduler is needed. This condition requires codesigning for the scheduler and the burst allocator.
- To satisfy traffic requirements such as real-time delay constraints, the burst allocator has to arrange bursts based on the traffic scheduling knowledge from the scheduler. For example, bursts for urgent real-time traffic should first be allocated to avoid packet dropping.
- Simplicity is a critical concern, because a frame is typically 5 ms [5], which means that the burst allocation scheme needs to be executed every 5 ms.

In the literature, prior studies design solely either the scheduler [6]–[10] or the burst allocator [4], [11]–[14] to address the reduction of IE overheads. Nevertheless, we point out the necessity of the cross-layer design by the following three reasons. First, the amount of IE overheads highly depends on the number of scheduled MSSs and the number of fragmented bursts, where prior work handles the two issues by the scheduler and the burst allocator, respectively. However, if we only consider either issue, the efficiency of overhead reduction is limited. Second, without considering burst arrangements, the scheduler may fail to satisfy MSSs' requirements, because extra IE overheads will occupy the limited frame space. Third, without considering the scheduling assignments, the burst allocator may kick out some important data of MSSs (due to out-of-frame space). This case may cause unfairness among MSSs and high packet-dropping ratios of real-time traffic. Therefore, it is necessary to codesign both the scheduler and the burst allocator due to their inseparable dependency.

In this paper, we propose a cross-layer framework that contains a *two-tier priority-based scheduler* and a *bucket-based burst allocator*. The scheduler assigns priorities to MSSs' traffic in a two-tier manner and allocates resources to the traffic based on its priority. In the first tier, traffic is differentiated by its type. Urgent real-time traffic is assigned with the highest level-1 priority to avoid its packets being dropped in the next frame. Then, a γ ratio $(0 < \gamma < 1)$ of nonurgent real-time traffic is assigned with level-2 priority, and non-real-time traffic is given level-3 priority. The aforementioned design has two advantages. First, we can avoid generating too many urgent real-time traffic in the next frame. Second, non-real-time traffic can have opportunity to be served to avoid being starved. In the second tier, traffic of the same type (i.e., the same priority level in the first tier) is assigned with different priorities calculated by the following four factors:

1) current transmission rates;
2) average transmission rates;
3) admitted data rates;
4) queue lengths.

The BS can have the knowledge of the aforementioned four factors, because all downlink traffic is queued in the BS, and MSSs will report their average channel qualities to the BS [15]. Unlike traditional priority-based solutions that are partial to non-real-time traffic [10], our novel two-tier priority scheduling scheme not only prevents urgent real-time traffic from incurring packet dropping (through the first tier) but maintains long-term fairness (through the second tier) as well. The network throughput is also improved by giving a higher priority to MSSs that use higher transmission rates (in the second tier). In addition, the scheduler can adjust the number of MSSs to be served and assign resources to traffic according to the burst arrangement manner (from the burst allocator) to significantly reduce IE overheads. This design is neglected in prior studies and has a significant impact on overhead reduction and network performance.

On the other hand, the burst allocator divides the free space of each downlink subframe into a special structure that consists of several "buckets" and then arranges bursts in a bucket-by-bucket manner. Given k requests to be filled in a subframe, we show that this burst allocation scheme generates at most k plus a small constant number of IEs. In addition, the burst allocator will arrange bursts according to the priority design from the scheduler so that the burst allocation can satisfy MSSs' traffic requirements. The aforementioned bucket-based design incurs very low computation complexity and can be implemented on most low-cost Worldwide Interoperability for Microwave Access (WiMAX) chips [16]. Explicitly, in our cross-layer framework, both the scheduler and the burst allocator are tightly coupled together to solve the problems of overhead reduction, real-time and non-real-time traffic scheduling, and burst allocation.

The major contributions of this paper are fourfold. First, we point out the necessity of codesigning both the scheduler and the burst allocator to improve network performance and propose a cross-layer framework that covers overhead reduction, real-time and non-real-time traffic scheduling, and burst allocation. Our framework is more complete and efficient than prior studies. Second, we develop a two-tier priority-based scheduler that distributes resources among MSSs according to their traffic types, transmission rates, and queue lengths. The proposed scheduler improves network throughput, guarantees traffic requirements, and maintains long-term fairness. Third, a low-complexity bucket-based scheme is designed for burst allocation, which significantly improves the utilization of downlink subframes. Fourth, we analyze the upper bound of the amount of IE overheads and the potential throughput degradation caused by the proposed burst allocator, which is used to validate the simulation experiments and provide guidelines for the setting of the burst allocator. Extensive simulations are also conducted, and their results validate that our cross-layer framework can achieve high network throughput, maintain long-term fairness, alleviate real-time traffic delays, and improve downlink subframe utilization.

The rest of this paper is organized as follows. Section II surveys the related work. Section III gives the problem formulation. The cross-layer framework is proposed in Section IV. Section V analyzes the expected network throughput of the proposed framework. Extensive simulation results are given in Section VI. Conclusions are drawn in Section VII.

II. Related Work

Most of the prior studies on resource allocation in IEEE 802.16 OFDMA networks implement either the scheduler or the burst allocator. For the implementation of the scheduler, the study in [6] proposes a scheduling scheme according to MSSs' signal-to-noise ratios (SNRs) to achieve rate maximization. The work in [7] proposes a utility function to evaluate the tradeoff between network throughput and long-term fairness. In [8], an opportunistic scheduler is proposed by adopting the instantaneous channel quality of each MSS to maintain fairness. However, these studies do not consider the delay requirements of real-time traffic. The work in [9] tries to minimize the blocking probability of MSSs' traffic requests, and thus, the packet-dropping ratios of real-time traffic may be reduced. Nevertheless, all of the aforementioned studies [6]–[9] do not address the issue of overhead reduction. The work in [10] tries to reduce IE overhead from the perspective of the scheduler, where the number of MSSs to be served in each subframe is reduced to avoid generating too many IEs. However, without the help of the burst allocator, the efficiency of overhead reduction becomes insignificant, and some important data (e.g., urgent real-time traffic) may not be allocated with bursts because of out-of-frame space. In this case, some MSSs may encounter serious packet dropping.

On the other hand, several studies consider implementing the burst allocator. The work in [17] proposes a new control message for periodic resource assignment to reduce duplicate signaling. Reference [18] suggests piggybacking IEs on data packets to increase the utilization of downlink subframes. However, both studies [17], [18] involve modifying the standard. The work in [4] proposes two heuristics for burst allocation: The first heuristic scans the free space in a downlink subframe row by row to try to fully utilize the space, but it may generate a large number of IEs. The second heuristic presegments a subframe into several rectangles, and a request will choose a rectangle larger than it for allocation; however, this scheme requires prior knowledge of the request distribution. The work in [11] first allocates bursts for large requests. Nevertheless, larger requests may not be necessarily more important or urgent. Several studies consider allocating bursts in a column-by-column manner. In [12], bursts with the same MCS are combined into a large one. However, this scheme is not compliant to the standard, because a burst may contain requests from multiple MSSs. The study in [13] pads zero bits in each column's end, which may cause low subframe utilization. The work in [14] adopts a backward columnwise allocation scheme, where the bursts are allocated from the right–down side to the left–up side of the subframe. However, this scheme requires $3n$ bursts for n MSSs in the worst case. As shown, existing research efforts may pad too many useless bits, generate too many IEs, or leave unused slot holes.

Few studies implement both the scheduler and the burst allocator, but they do not consider reducing the IE overhead. The studies in [19] and [20] try to arrange resources to MSSs to maximize their data rates and maintain fairness. However, they do not consider the delay requirements of real-time traffic. The studies in [21] and [22] develop a one-tier priority-based scheduler to allocate resources to each MSS to exactly satisfy its demand. Thus, the delay requirement of real-time traffic could be guaranteed, but the network throughput may be degraded. Nevertheless, all of the studies [19]–[22] neglect the issue of overhead reduction, which may lead to low subframe utilization and low network throughput. We will show by simulations in Section VI that, without reducing the IE overhead, the quality-of-service (QoS) requirements of MSSs' traffic may not be satisfied, particularly when the network becomes saturated.

Table I compares the features of prior studies and our cross-layer framework. It is shown that our cross-layer framework

TABLE I
COMPARISON OF PRIOR WORK AND OUR CROSS-LAYER FRAMEWORK

features	network throughput	long-term fairness	rate satisfaction[‡]	real-time traffic	subframe utilization	burst allocation complexity
references [6]–[8]	√	√	√			N/A
reference [9]	√	√	√	√		N/A
reference [10]	√	√	√		√	N/A
references [4], [12], [14]					√	$O(n), O(n), O(n^2)$
references [11], [13]				√	√	$O(n)$
reference [19], [20]	√	√	√			$O(n^2), O(n)$
references [21], [22]	√		√	√		$O(n)$
our framework	√	√	√	√	√	$O(n)$

[†] n is the number of MSSs.
[‡] The rate satisfaction is to evaluate the degree of starvation of non-real-time traffics.

covers all of the features. In addition, our cross-layer framework has the least computation complexity in burst allocation. Thus, it can be implemented on most WiMAX low-cost chips.

III. RESOURCE ALLOCATION PROBLEM

We consider the downlink communication in an IEEE 802.16 OFDMA network using the TDD mode. The mandatory PUSC model is adopted so that there is no issue of subchannel diversity (because MSSs will report only their average channel qualities to the BS, as mentioned in Section I). The BS supports multiple MSSs in a point-to-multipoint manner, where each MSS has its admitted real-time and non-real-time traffic rates. The BS has to arrange the radio resource to the MSSs according to their traffic demands.

The radio resource is divided into frames, where each frame is further divided into a downlink subframe and an uplink subframe (see Fig. 1). A downlink subframe is modeled by a 2-D array with X time units (in the time domain) and Y subchannels (in the frequency domain). The basic unit in the $X \times Y$ array is called a *subchannel time slot* (or simply a *slot*). Each downlink subframe is composed of the following three portions: 1) *preamble*; 2) *control*; and 3) data. The control portion contains a DL-MAP and an uplink map (UL-MAP) to indicate the downlink and uplink resource allocation in the current frame, respectively. The downlink allocation unit is a subarray, called a *downlink burst* (or simply a *burst*), in the $X \times Y$ array. Each burst is denoted by (x, y, w, h), where x is the starting time unit, y is the starting subchannel, w is the burst's width, and h is the burst's height. An MSS can own more than one burst in a subframe. However, no two bursts can overlap with each other. Fig. 1 gives some examples. Bursts 1 and 2 can coexist, but bursts 2 and 3 cannot coexist. Each burst requires one IE in the DL-MAP to describe its size and location in the subframe. According to the standard, each burst carries the data of exact one MSS. Explicitly, from the scheduler's perspective, the number of bursts (and, thus, IEs) will increase when more MSSs are scheduled. On the other hand, from the burst allocator's perspective, more IEs are required when an MSS's data are distributed over multiple bursts. An IE requires 60 b encoded by the QPSK 1/2 MCS [5]. Because each slot can carry 48 b by QPSK 1/2, an IE occupies 5/4 slots, which has a significant impact on the available space to allocate bursts in a downlink subframe.

The resource allocation problem is formulated as follows. There are n MSSs in the network, where each MSS M_i, $i = 1, \ldots, n$, is admitted with an average real-time data rate of R_i^{rt} (in bits/frame) and a minimal non-real-time data rate of R_i^{nrt} (in bits/frame). Let C_i be the current transmission rate[1] (in bits/slot) for the BS to send data to M_i, which may change over frames. The objective is to design a cross-layer framework that contains both the scheduler and the burst allocator to arrange bursts to MSSs such that we can reduce the IE overhead, improve the network throughput, achieve long-term fairness, alleviate real-time traffic delays, and maximally utilize downlink subframes. In addition, the design of the cross-layer framework should not be very complicated so that it can execute within a frame duration (i.e., 5 ms) and be implemented in most low-cost WiMAX chips. Note that the *fairness index* (FI) in [23] is adopted to evaluate the long-term fairness of a scheme as follows:

$$FI = \frac{\left(\sum_{i=1}^{n} SD_i\right)^2}{n \sum_{i=1}^{n} (SD_i)^2}$$

where SD_i is the *share degree* of M_i, which is calculated by

$$SD_i = \frac{\sum_{j=0}^{T-1} \left(\tilde{A}_i^{rt}(f_c - j) + \tilde{A}_i^{nrt}(f_c - j)\right)}{T \times \left(R_i^{rt} + R_i^{nrt}\right)} \tag{1}$$

where $\tilde{A}_i^{rt}(x)$ and $\tilde{A}_i^{nrt}(x)$ are the amounts of real-time and non-real-time traffic allocated to M_i in the xth frame, respectively, f_c is the current frame index, and T is the window size (in frames), over which we measure fairness. We denote by $U_d(x)$ the *utilization* of the xth downlink subframe, which is defined by the ratio of the number of slots used to transmit data to $X \times Y$. Thus, the average downlink utilization over T frames is $\sum_{j=0}^{T-1} U_d(f_c - j)/T$. Table II summarizes the notations used in this paper.

IV. PROPOSED CROSS-LAYER FRAMEWORK

Fig. 2 shows the system architecture of our cross-layer framework, which is composed of the following two components: 1) the *two-tier priority-based scheduler* and 2) the

[1]The estimation of the transmission rate highly depends on the path loss, fading, and propagation model. Here, we assume that the BS can accurately estimate the transmission rate for each MSS and will discuss how we can conduct the estimation in Section VI.

TABLE II
SUMMARY OF NOTATIONS

notation	definition
n	the number of admitted MSSs in the network
X	the number of units in time domain of a downlink subframe
Y	the number of subchannels in frequency domain of a downlink subframe
FS	the free-space (in slots) in a downlink subframe
T	the window size (in frames)
Δ_{bkt}	the bucket size (in slots)
C_i	the current transmission rate (in bits/slot) for the BS to send data to MSS M_i
C_i^{avg}	the average transmission rate (in bits/slot) for the BS to send data to M_i in recent T frames
R_i^{rt}/R_i^{nrt}	the admitted data rate (in bits/frame) of M_i's real-time/non-real-time traffic
B_i^{rt}/B_i^{nrt}	the amount of real-time/non-real-time queued data (in bits) of M_i
Q_i^{rt}/Q_i^{nrt}	M_i's real-time/non-real-time resource assignments (in bits) generated by the scheduler
A_i^{rt}/A_i^{nrt}	the amount of real-time/non-real-time data (in bits) allocated to M_i by the burst allocator
I_i^{rt}/I_i^{nrt}	M_i's importance factors to allocate real-time/non-real-time traffics
S_i^{nrt}	the non-real-time rate satisfaction ratio of M_i in recent T frames
θ_{IE}	the size of an IE (in slots)
B	the number of buckets in a downlink subframe

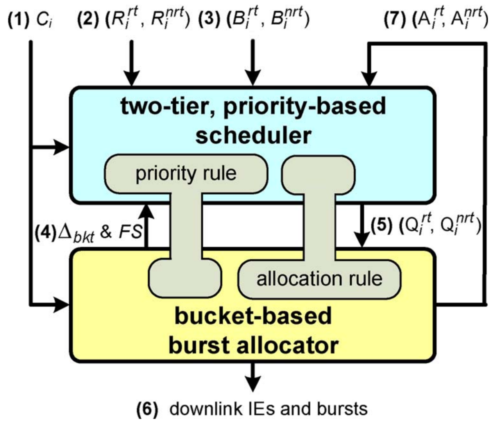

(1) C_i
(2) (R_i^{rt}, R_i^{nrt})
(3) (B_i^{rt}, B_i^{nrt})
(7) (A_i^{rt}, A_i^{nrt})
two-tier, priority-based scheduler
priority rule
(4)Δ_{bkt} & FS
(5) (Q_i^{rt}, Q_i^{nrt})
allocation rule
bucket-based burst allocator
(6) downlink IEs and bursts

Fig. 2. System architecture of the proposed cross-layer framework, where $i = 1 \ldots n$.

bucket-based burst allocator. The transmission rate C_i for each MSS M_i (see Fig. 2, label 1) is periodically reported to the scheduler and the burst allocator. Each M_i's admitted rates R_i^{rt} and R_i^{nrt} (see Fig. 2, label 2) are sent to the scheduler when M_i first associates with the BS or when R_i^{rt} and R_i^{nrt} change. The scheduler also monitors the current amounts of queued real-time and non-real-time data B_i^{rt} and B_i^{nrt} (see Fig. 2, label 3). The burst allocator informs the scheduler of the bucket size Δ_{bkt} and the available *free-space FS* in the current downlink subframe (see Fig. 2, label 4) to help the scheduler distribute resources among MSSs' traffic, where

$$FS = X \times Y - (\text{FCH size}) - (\text{UL-MAP size}) - (\text{size of DL-MAP control fields}) \quad (2)$$

where FCH is the frame control header. The UL-MAP size can be known in advance, because the uplink subframe is allocated before the downlink subframe. The DL-MAP control fields contain all parts of DL-MAP, except for IEs, which are yet to be decided by the burst allocator. The scheduler's mission is to generate each M_i's real-time and non-real-time resource assignments Q_i^{rt} and Q_i^{nrt} (see Fig. 2, label 5) to the burst allocator. Based on Q_i^{rt} and Q_i^{nrt}, the burst allocator arranges IEs and bursts to each M_i (see Fig. 2, label 6). The actual real-time and non-real-time traffic allocated to M_i are written, respectively, as A_i^{rt} and A_i^{nrt} (see Fig. 2, label 7) and are fed to the scheduler for future scheduling.

In our cross-layer framework, the *priority rule* defined in the scheduler helps the burst allocator determine how bursts can be arranged for MSSs' traffic. On the other hand, the *allocation rule* defined in the burst allocator also helps the scheduler to determine how resources can be assigned to MSSs' traffic. Both the priority and allocation rules are similar to tenons in the cross-layer framework, which make the scheduler and the burst allocator tightly cooperate with each other.

Due to the NP-complete nature of the burst allocation problem and the hardware constraints of low-cost WiMAX chips, it is inefficient and yet infeasible to derive an optimal solution for arranging IEs and bursts in a short frame duration. Therefore, to keep our burst allocator simple and efficient, we adopt a *bucket* concept as follows. The available free-space FS in the current subframe is horizontally sliced into a number of buckets, each of size Δ_{bkt} (see Fig. 3 for an example). The size Δ_{bkt} serves as the allocation unit in our scheme. As shown, the scheduler always keeps $(Q_i^{rt} + Q_i^{nrt})$ as a multiple of Δ_{bkt} for each M_i. This way, the burst allocator can easily arrange bursts in a "bucket-by-bucket" manner, well utilize the frame resource, and generate quite few bursts and, thus, IEs (which will be proved to have an upper bound in Section IV-B). In addition, the long-term fairness is achieved, because the actual allocation (A_i^{rt}, A_i^{nrt}) by the burst allocator is likely to be quite close to the assignment (Q_i^{rt}, Q_i^{nrt}) by the scheduler for each $i = 1, \ldots, n$.

A. *Two-Tier Priority-Based Scheduler*

In each frame, the scheduler will generate resource assignments (Q_i^{rt}, Q_i^{nrt}), $i = 1, \ldots, n$ to the burst allocator. To generate these assignments, the scheduler adopts a two-tier priority rule. In the first tier, traffic is differentiated by its type and is given priority levels according to the following order.

– **P1**: urgent real-time traffic whose packets will pass their deadlines at the end of this frame;
– **P2**: real-time traffic ranked top γ ratio $(0 < \gamma < 1)$ sorted by their importance;
– **P3**: non-real-time traffic sorted by their importance.

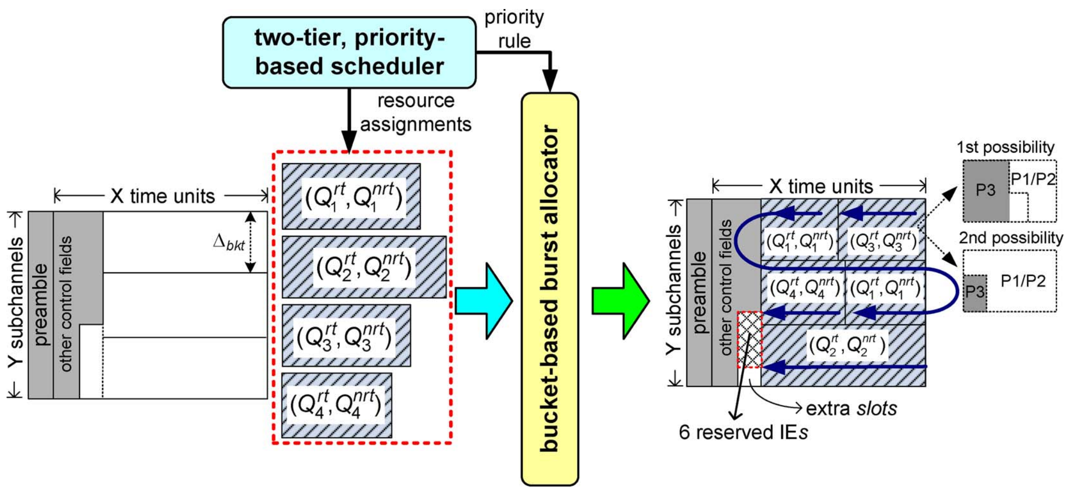


Fig. 3. Example of the bucket-based burst allocation with three buckets and four resource assignments.

Then, in the second tier, traffic of the same type is assigned with different priorities by its *importance*, which is calculated by the following four factors:

1) current transmission rates;
2) average transmission rates;
3) admitted data rates;
4) queue lengths.

In particular, for priority level **P2**, we rank the *importance* of M_i's real-time traffic by

$$I_i^{rt} = C_i \times \frac{C_i}{C_i^{\text{avg}}} \times \frac{B_i^{rt}}{R_i^{rt}}. \tag{3}$$

Here, the importance I_i^{rt} involves the following three factors that are multiplied together.

1) A higher transmission rate C_i gives M_i a higher rating to improve the network throughput.
2) A higher ratio C_i/C_i^{avg} gives M_i a higher rating to prevent starvation for MSSs with low average rates, where C_i^{avg} is the average transmission rate for the BS to send data to M_i in the most recent T frames. In particular, supposing that an MSS encounters a bad channel condition for a long period (i.e., a lower C_i^{avg} value), we still prefer this MSS if it can now enjoy a higher transmission rate (i.e., $C_i > C_i^{\text{avg}}$). In addition, a higher C_i/C_i^{avg} value means that the MSSs is currently in a better condition; therefore, we give it a higher priority to improve the potential throughput.
3) A higher ratio B_i^{rt}/R_i^{rt} gives M_i a higher rating to favor MSSs with more backlogs.

Similarly, for priority level **P3**, we rank the importance of M_i's non-real-time traffic by

$$I_i^{nrt} = C_i \times \frac{C_i}{C_i^{\text{avg}}} \times \frac{1}{S_i^{nrt}} \tag{4}$$

where S_i^{nrt} is the *non-real-time rate satisfaction ratio* of M_i in the most recent T frames, which is calculated by

$$S_i^{nrt} = \frac{\sum_{j=0}^{T-1} A_i^{nrt}(f_c - j)}{T \times R_i^{nrt}}. \tag{5}$$

A small S_i^{nrt} means that M_i's non-real-time traffic may be starved. Thus, a smaller S_i^{nrt} gives M_i a higher rating.

The aforementioned two-tier priority rule not only prevents urgent real-time traffic from incurring packet dropping (through the first tier) but maintains long-term fairness (through the second tier) as well. The network throughput is also improved by giving a higher priority to MSSs that use higher transmission rates (in the second tier). In addition, by giving a γ ratio of nonurgent real-time traffic with level-2 priority, the amount of urgent real-time traffic in the next frame can be reduced, and non-real-time traffic can have opportunity to send their data.

In the following procedure, we present the detailed operations of our scheduler. Let e_i be a binary flag to indicate whether an IE has been allocated for M_i, $i = 1, \ldots, n$. Initially, we set all $e_i = 0$, $i = 1, \ldots, n$. In addition, the free space FS is deducted by $((Y/\Delta_{bkt}) - 1) \times \theta_{IE}$ to preserve the space for potential IEs caused by the burst allocator (this case will be discussed in the next section), where $\theta_{IE} = 5/4$ is the size of an IE.

1) Let U_i^{rt} be the amount of data of M_i's urgent real-time traffic in the current frame. For all M_i with $U_i^{rt} > 0$, we sort them according to their C_i values in a descending order. Then, we schedule the free-space FS for each of them as follows until $FS \leq 0$.
   a) Reserve an IE for M_i by setting $FS = FS - \theta_{IE}$. Then, set $e_i = 1$.
   b) If $FS > 0$, assign resource $Q_i^{rt} = \min\{FS \times C_i, U_i^{rt}\}$ to M_i and set $FS = FS - \lceil Q_i^{rt}/C_i \rceil$. Then, deduct Q_i^{rt} from B_i^{rt}.
2) After step 1, if $FS > 0$, we sort all M_i that have real-time traffic according to their I_i^{rt} values [by (3)]. Then, we schedule the resource for each of them as follows until either all MSSs in the top γ ratio are examined or $FS \leq 0$.
   a) If $e_i = 0$, reserve an IE for M_i by setting $FS = FS - \theta_{IE}$ and $e_i = 1$.
   b) If $FS > 0$, assign more resources $\delta = \min\{FS \times C_i, B_i^{rt}\}$ to M_i. Then, set $Q_i^{rt} = Q_i^{rt} + \delta$ and $FS = FS - \lceil \delta/C_i \rceil$. Deduct δ from B_i^{rt}.

3) After step 2, if $FS > 0$, we sort all M_i according to their I_i^{nrt} values [by (4)]. Then, we schedule the resource for each of them as follows until either all MSSs are examined or $FS \leq 0$.
   a) If $e_i = 0$, reserve an IE for M_i by setting $FS = FS - \theta_{IE}$ and $e_i = 1$.
   b) If $FS > 0$, assign more resources $\delta = \min\{FS \times C_i, B_i^{nrt}\}$ to M_i. Then, set $Q_i^{nrt} = \delta$ and $FS = FS - \lceil \delta / C_i \rceil$. Deduct δ from B_i^{nrt}.
4) Because the bucket size Δ_{bkt} is the allocation unit in our burst allocator, in this step, we will do a fine-tuning on Q_i^{rt} and Q_i^{nrt} such that $(Q_i^{rt} + Q_i^{nrt})$ is aligned to a multiple of Δ_{bkt} for each M_i. To do so, we will gradually *remove* some slots from Q_i^{nrt} and then Q_i^{rt} until $(((Q_i^{rt} + Q_i^{nrt})/C_i) \bmod \Delta_{bkt}) = 0$. One exception is when much of the data in Q_i^{rt} are urgent, which makes removing any resource from M_i impossible. In this case, we will *add* more slots to M_i until $(((Q_i^{rt} + Q_i^{nrt})/C_i) \bmod \Delta_{bkt}) = 0$. The aforementioned adjustment (i.e., removal and addition) may make the total resource assignment below or beyond the available resource FS. If so, we will further remove some slots from the MSSs with less importance or add some slots to the MSSs with more importance until the total resource assignment is equal to the initial free space given by the burst allocator.

Fig. 4 illustrates the flowchart of the scheduler. To summarize, our scheduler generates the resource assignment according to the following three priorities: **P1**) urgent traffic; **P2**) real-time traffic; and **P3**) non-real-time traffic. Step 1 first schedules MSSs with urgent traffic to alleviate their real-time traffic delays. Step 2 schedules the top γ ratio of MSSs to reduce the number of MSSs that may have urgent traffic in the following frames. This step also helps reduce the IE overhead of future frames caused by urgent traffic, which is neglected by prior studies. Step 3 schedules MSSs with lower non-real-time satisfaction ratios to prevent them from starvation. Finally, step 4 reshapes all assignments such that each $(Q_i^{rt} + Q_i^{nrt})$ is divisible by Δ_{bkt}. This step will help the burst allocator fully utilize a downlink subframe.

We then analyze the time complexity of our scheduler. In step 1, sorting MSSs by their C_i values takes $O(n \lg n)$ time, and scheduling the resources for the MSSs with urgent traffic takes $O(n)$ time. In step 2, sorting MSSs by their I_i^{rt} values requires $O(n \lg n)$ time, and scheduling the resources for the top γ ratio of MSSs requires at most $O(\gamma n)$ time. In step 3, sorting MSSs by their I_i^{nrt} values costs $O(n \lg n)$ time, and scheduling the resources for the MSSs with non-real-time traffic takes $O(n)$ time. In step 4, reshaping all requests spends at most $O(n)$ time. Thus, the total time complexity is $O(n \lg n + n + n \lg n + \gamma n + n \lg n + n + n) = O(n \lg n)$.

B. Bucket-Based Burst Allocator

Ideally, the free space FS in (2) should accommodate each resource assignment (Q_i^{rt}, Q_i^{nrt}) calculated by the scheduler and its corresponding IE(s). However, because the burst allocation problem is NP-complete, our bucket-based heuristic will try to squeeze as more MSSs' assignments into FS as possible and allocate one burst per assignment with a very high possibility. If more than one burst is required, more IEs are needed, in which case, some assignments that were originally arranged by the scheduler may be trimmed down or even kicked out by the burst allocator. Given the free space FS by (2), bucket size Δ_{bkt}, and assignments (Q_i^{rt}, Q_i^{nrt})'s from the scheduler, our bucket-based heuristic works as follows.

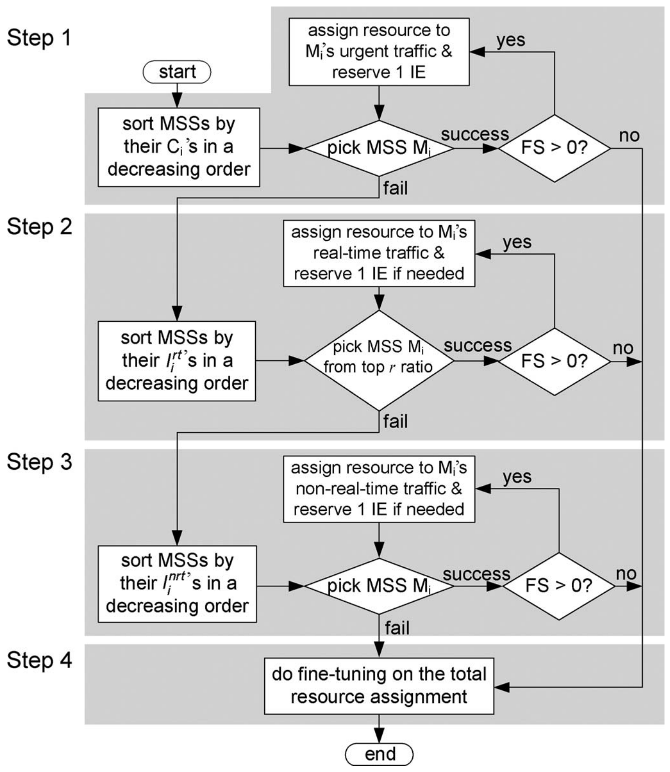

Step 1
start
sort MSSs by their C_i's in a decreasing order
pick MSS M_i
success
FS > 0?
yes
no
assign resource to M_i's urgent traffic & reserve 1 IE
fail
Step 2
sort MSSs by their I_i^rt's in a decreasing order
pick MSS M_i from top r ratio
assign resource to M_i's real-time traffic & reserve 1 IE if needed
Step 3
sort MSSs by their I_i^nrt's in a decreasing order
pick MSS M_i
assign resource to M_i's non-real-time traffic & reserve 1 IE if needed
Step 4
do fine-tuning on the total resource assignment
end

Fig. 4. Flowchart of the two-tier priority-based scheduler.

1) Horizontally[2] slice FS into Y/Δ_{bkt} buckets, each of a height Δ_{bkt}, where Y is divisible by Δ_{bkt}. Fig. 3 shows an example by slicing FS into three buckets.
2) Let k be the number of resource assignments given by the scheduler. We reserve $\lceil (k + (Y/\Delta_{bkt}) - 1) \times \theta_{IE} \rceil$ slots for IEs at the left side of the subframe. In fact, the scheduler has also reserved the space for these IEs, and its purpose will become clear later on. Fig. 3 gives an example. Because there are four assignments, $4 + 3 - 1$ IEs are reserved.
3) We then assign bursts to satisfy these resource assignments according to their priorities originally defined in the scheduler. Because each assignment (Q_i^{rt}, Q_i^{nrt}) may have data mixed in the categories of **P1**, **P2**, and **P3**, we redefine its priority as follows.
   a) An assignment with data in **P1** has a higher priority than an assignment without data in **P1**.
   b) Without the existence of data in **P1**, an assignment with data in **P2** has a higher priority than an assignment without data in **P2**.

[2]We can also vertically slice FS, but the effect will be the same.

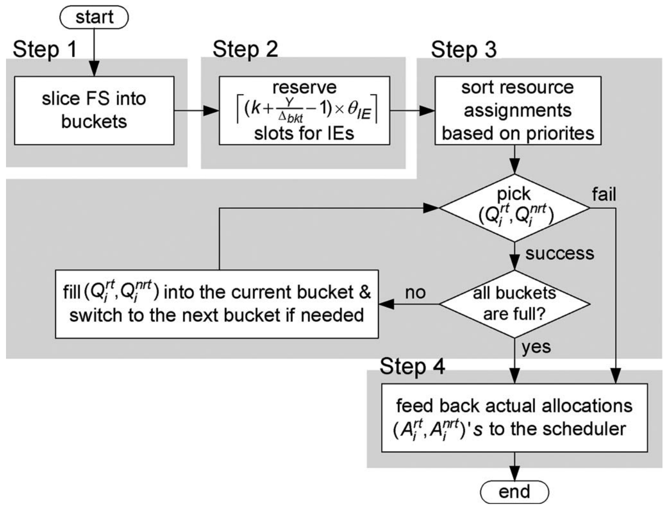

start
Step 1
slice FS into buckets
Step 2
reserve $\lceil (k+\frac{Y}{\Delta_{bkt}}-1)\times\theta_{IE}\rceil$ slots for IEs
Step 3
sort resource assignments based on priorites
pick (Q_i^{rt},Q_i^{nrt})
fail
success
all buckets are full?
no
fill (Q_i^{rt},Q_i^{nrt}) into the current bucket & switch to the next bucket if needed
yes
Step 4
feed back actual allocations (A_i^{rt},A_i^{nrt})'s to the scheduler
end

Fig. 5. Flowchart of the bucket-based burst allocator.

Then, bursts are allocated in a bucket-by-bucket manner. In particular, when an assignment (Q_i^{rt}, Q_i^{nrt}) is examined, it will be placed starting from the previous stop point and fill up the bucket from right to left until either (Q_i^{rt}, Q_i^{nrt}) is satisfied or the left end of the bucket is encountered. In the latter case, we will move to the right end of the next bucket and repeat the aforementioned allocation process. In addition, this "cross-bucket" behavior will require one extra IE for the request. The aforementioned operation is repeated until either all assignments are examined or all buckets are exhausted. Fig. 3 gives one example, where the four assignments are prioritized by $(Q_3^{rt}, Q_3^{nrt}) > (Q_1^{rt}, Q_1^{nrt}) > (Q_4^{rt}, Q_4^{nrt}) > (Q_2^{rt}, Q_2^{nrt})$. Assignment (Q_1^{rt}, Q_1^{nrt}) requires two IEs, because it involves one cross-bucket behavior.

4) According to the allocation in step 3, we place each resource assignment (Q_i^{rt}, Q_i^{nrt}) into its burst(s). In addition, the amount of actual allocation is written into each (A_i^{rt}, A_i^{nrt}) and fed back to the scheduler for future scheduling.

Fig. 5 illustrates the flowchart of the burst allocator. We make some remarks as follows. First, because there are Y/Δ_{bkt} buckets, there are at most $((Y/\Delta_{bkt}) - 1)$ *cross-bucket* burst assignments, and thus, at most $((Y/\Delta_{bkt}) - 1)$ extra IEs are needed. To accommodate this need, some assignments may slightly be trimmed down. Thus, (Q_i^{rt}, Q_i^{nrt}) and (A_i^{rt}, A_i^{nrt}) are not necessarily the same. However, the difference should be very small. Second, the bucket that is located at the boundary of reserved IEs and data (e.g., the third bucket in Fig. 3) may have some extra slots (e.g., the lower left corner of the third bucket). These extra slots are ignored in the aforementioned process for ease of presentation, but they can be used to allocate bursts to further improve space efficiency. Third, because each cross-bucket behavior will require one extra IE and there are Y/Δ_{bkt} buckets, the number of IEs required is bounded, as proved in Theorem 1.

Theorem 1: In the bucket-based burst allocator, the $(k + (Y/\Delta_{bkt}) - 1)$ IEs reserved in step 2 are sufficient for the burst allocation in step 3.

Proof: Given Y/Δ_{bkt} buckets $\hat{b}_1, \hat{b}_2, \ldots,$ and $\hat{b}_{Y/\Delta_{bkt}}$, we can concatenate them into one virtual bucket $\hat{b}$ with $((Y/\Delta_{bkt}) - 1)$ joints. We then allocate one virtual burst for each request from the scheduler in $\hat{b}$; therefore, we have at most k virtual bursts. Then, we replace each virtual burst by one real burst. However, we require one extra real burst whenever the replaced virtual burst crosses one joint. The worst case occurs when each of the $((Y/\Delta_{bkt}) - 1)$ joints is crossed by one virtual burst. In this case, we require $(k + (Y/\Delta_{bkt}) - 1)$ real bursts to replace all virtual bursts. Because each real burst requires one IE, we have to reserve at most $(k + (Y/\Delta_{bkt}) - 1)$ IEs. ■

In comparison, a naive burst allocation will require the worst case of $3k$ IEs if the allocation goes in a row-major or column-major way [14] (because each request may require up to three IEs). In our scheme, the bucket size Δ_{bkt} can dynamically be adjusted to reflect the "grain size" of our allocation. A larger grain size may cause fewer IEs but sacrifice resource utilization, whereas a smaller grain size may cause more IEs but improve resource utilization. We will discuss the effect of Δ_{bkt} in Section VI-F.

We then analyze the time complexity of our burst allocator. Because we allocate bursts in a zigzag manner, the time complexity is proportional to the number of bursts. By Theorem 1, we have at most $(k + (Y/\Delta_{bkt}) - 1)$ bursts. Because we have $k \leq n$ and Y/Δ_{bkt} is usually smaller than n, the time complexity is $O(k + (Y/\Delta_{bkt}) - 1) = O(n)$.

To conclude, the proposed scheduler and burst allocator are dependent on each other by the following two designs. First, the scheduler reserves the extra IE space caused by the bucket partition and arranges resources to MSSs' traffic so that the resource assignments can align to buckets. Thus, we can enhance the possibility that the burst allocator fully satisfies the resource assignments from the scheduler. Second, the burst allocator follows the priority rule in the scheduler to arrange bursts. Thus, even if the frame space is not enough to satisfy all traffic, urgent real-time traffic can still be arranged with bursts to catch their approaching deadlines.

V. Analysis of Network Throughput Loss by the Bucket-Based Scheme

Given an ideal scheduler, we analyze the loss of network throughput caused by our bucket-based burst allocator. To simplify the analysis, we assume that the network has only traffic of priority levels **P1** and **P3** and each MSS has infinite data in **P3**. (Traffic of **P2** will eventually become urgent traffic of **P1**.) Then, we calculate the difference between the expected throughput by our burst allocator and the maximum throughput by an *ideal* burst allocator. In the ideal burst allocator, the number of IEs is equal to the number of resource assignments from the scheduler. In addition, the frame resource is always first allocated to urgent traffic (**P1**) and then to non-real-time traffic (**P3**) with the highest transmission rate. It follows that the following two factors may degrade the network throughput by our burst allocator: 1) extra IEs incurred by step 3 in Section IV-B and 2) the data padding of low-rate non-real-time traffic at the boundary between the data in **P1** and **P3**. In particular, each burst must begin with the data in **P1**, followed by the data in **P3**. Furthermore, if the data in **P3** covers more than one column, it must be sent at the highest transmission rate. If the data in **P3** covers less than a column, it may be sent at a nonhighest transmission rate. The right-hand side of Fig. 3 shows these two possibilities, where **P2** is empty. Note that, in the first possibility, all data in **P3** must be transmitted at the highest rate; otherwise, the shaded area will be allocated to the data in **P3** of other MSSs using the highest rate.

Following the aforementioned formulation, our objective is to find the throughput loss $\mathcal{L}$ by our burst allocator compared with the ideal burst allocator as

$$\mathcal{L} = E[\tilde{\mathcal{O}}] \times c_{\text{high}} + E[\tilde{\mathcal{S}}] \tag{6}$$

where $\tilde{\mathcal{O}}$ is the random variable that represents the number of extra IEs caused by buckets, and $\tilde{\mathcal{S}}$ is the random variable that represents the throughput degradation (in bits) caused by the low-rate padding in the shaded area of the second possibility on the right-hand side of Fig. 3. To simplify the analysis, we assume that there are only two transmission rates c_{high} and c_{low}, where $c_{\text{high}} > c_{\text{low}}$. The probability that an MSS is in either rate is equal.

A. Calculation of $E[\tilde{\mathcal{O}}]$

We first give an example to show how our analysis works. Suppose that we have three MSSs and three buckets. Each bucket has two *arrangement units*, each having Δ_{bkt} slots. Thus, there are, in total, six arrangement units, denoted by O_1, O_2, O_3, O_4, O_5, and O_6. Resources that are allocated to the three MSSs can be represented by two separators "|." For example, we list the following three possible allocations: 1) $O_1O_2|O_3O_4|O_5O_6$; 2) $O_1O_2O_3O_4\|O_5O_6$; and 3) $O_1|O_2O_3O_4O_5|O_6$. In arrangement 1, we need no extra IE. In arrangement 2, MSS 2 receives no resource, but MSS 1 needs one extra IE. In arrangement 3, MSS 2 requires two IEs.

We will use arrangement units and separators to conduct the analysis. Suppose that we have n MSSs, $(Y/\Delta_{bkt})(= B)$ number of buckets, and $X \times B(= \alpha)$ arrangement units (i.e., each bucket has X arrangement units). This approach can be represented by arbitrarily placing $(n-1)$ separators along a sequence of α arrangement units. Bucket boundaries appear after each ith arrangement unit such that i is a multiple of X. Note that only $(B-1)$ bucket boundaries can cause extra IEs, as mentioned in Section IV. Whenever no separator appears at a bucket boundary, one extra IE is needed. There are, in total, $(\alpha + (n-1))!/(\alpha!(n-1)!)$ ways of placing these separators. Let $\tilde{\mathcal{E}}$ be the random variable that represents the number of bucket boundaries, where each of bucket boundaries is inserted by at least one separator. The probability of $(\tilde{\mathcal{E}} = e)$ is calculated by

$$Prob[\tilde{\mathcal{E}} = e] = \frac{C_e^{B-1} \times \frac{(\alpha-(B-1-e)+(n-1-e))!}{(\alpha-(B-1-e))!(n-1-e)!}}{\frac{(\alpha+(n-1))!}{\alpha!(n-1)!}}. \tag{7}$$

Note that the term C_e^{B-1} refers to the combinations of choosing e boundaries from the $(B-1)$ bucket boundaries. Each of these e boundaries is inserted by at least one separator. The remaining $(B-1-e)$ bucket boundaries must not be inserted by any separator. To understand the second term in the numerator of (7), we can denote by x_0 the number of separators before the first arrangement unit and by x_i the number of separators after the ith arrangement unit, $i = 1, \ldots, \alpha$. Explicitly, we have

$$x_0 + x_1 + \cdots + x_\alpha = n - 1 \quad \forall x_i \in \{0, 1, 2, \cdots\}.$$

However, when $\tilde{\mathcal{E}} = e$, $(B-1-e)$ of these x_i's must be 0. In addition, e of these x_i's must be larger than or equal to 1. Then, this problem is equivalent to finding the number of combinations of

$$y_0 + y_1 + \cdots + y_j + \cdots + y_{\alpha-(B-1-e)} = n - 1 - e \quad \forall y_j \in \{0, 1, 2, \ldots\}.$$

It follows that there are $(\alpha - (B-1-e) + (n-1-e))!/((\alpha - (B-1-e))!(n-1-e)!)$ combinations. Therefore, $E[\tilde{\mathcal{O}}]$ can be obtained by

$$\begin{aligned} E[\tilde{\mathcal{O}}] &= \sum_{e=0}^{B-1} (\text{number of extra IEs when } \tilde{\mathcal{E}} = e) \times Prob[\tilde{\mathcal{E}} = e] \\ &= \sum_{e=0}^{B-1} (B-1-e) \times \frac{C_e^{B-1} \times \frac{(\alpha-(B-1-e)+(n-1-e))!}{(\alpha-(B-1-e))!(n-1-e)!}}{\frac{(\alpha+(n-1))!}{\alpha!(n-1)!}}. \end{aligned} \tag{8}$$

B. Calculation of $E[\tilde{S}]$

Recall that $E[\tilde{S}]$ is the expected throughput degradation caused by the transmission of a burst at a low rate and the burst contains some data padding of non-real-time traffic. To calculate $E[\tilde{S}]$, let us define $\tilde{N}_L$ as the random variable of the number of MSSs using the low transmission rate c_{low}. Because there is no throughput degradation by MSSs using the high transmission rate c_{high}, the overall expected throughput degradation is

$$E[\tilde{S}] = \sum_{m=1}^{n} E[\tilde{S}|\tilde{N}_L = m] \times Prob[\tilde{N}_L = m]. \tag{9}$$

Let $\tilde{U}_i$ be the random variable that represents the amount of data of M_i's urgent traffic $i = 1, \ldots, n$. Here, we assume that $\tilde{U}_i$ is uniformly distributed among $[1, \mathcal{R}]$, where $\mathcal{R} \in \mathbb{N}$. Let $\tilde{X}_j^L$ be the random variable that represents the amount of throughput degradation (in bits) due to the data padding of M_j's non-real-time traffic when using c_{low}. Because the throughput degradation caused by MSSs using c_{high} is zero, we have

$$E[\tilde{S}|\tilde{N}_L = m] = E\left[\sum_{j=1}^{m} \tilde{X}_j^L\right]. \tag{10}$$

Explicitly, $\tilde{X}_i^L$ and $\tilde{X}_j^L$ are independent of each other for any $i \neq j$; therefore, we have

$$E\left[\sum_{j=1}^{m} \tilde{X}_j^L\right] = \sum_{j=1}^{m} E\left[\tilde{X}_j^L\right]. \tag{11}$$

Now, let us define I_i^U as an indicator of representing whether M_i has urgent traffic such that $I_i^U = 1$ if M_i has urgent traffic; otherwise, $I_i^U = 0$. Because the bursts of low-rate MSSs without urgent traffic will not contain the data padding of non-real-time traffic, no throughput degradation will be caused by them. Therefore, we can derive

$$\begin{aligned} E\left[\tilde{X}_j^L\right] &= E\left[\tilde{X}_j^L | I_j^U = 1\right] \times Prob\left[I_j^U = 1\right] \\ &= \left(\sum_{u=1}^{\mathcal{R}} \frac{f(\tilde{U}_j = u)}{\mathcal{R}}\right) \times Prob\left[I_j^U = 1\right] \qquad (12) \end{aligned}$$

where

$$f(\tilde{U}_j = u) = \left(\left\lceil \frac{u}{\Delta_{bkt} \times c_{\text{low}}} \right\rceil - \frac{u}{\Delta_{bkt} \times c_{\text{low}}}\right) \times \Delta_{bkt} \times (c_{\text{high}} - c_{\text{low}})$$

is a function for representing the throughput degradation caused by a low-rate MSS with non-real-time data padding when $\tilde{U}_j = u$.

By combining (9)–(12), we can derive that

$$E[\tilde{S}] = \sum_{m=1}^{n} \left(\sum_{j=1}^{m} \left(\sum_{u=1}^{\mathcal{R}} \frac{f(\tilde{U}_j = u)}{\mathcal{R}}\right) \times Prob\left[I_j^U = 1\right]\right) \times Prob[\tilde{N}_L = m]. \qquad (13)$$

Finally, the throughput loss by our burst allocator can be calculated by combining (8) and (13) into (6).

VI. Performance Evaluation

To verify the effectiveness of our cross-layer framework, we develop a simulator in C++ based on the architecture in [15], as shown in Fig. 6. The simulator contains three layers: The *traffic-generating module* in the upper layer creates the MSSs' demands according to their real-time and non-real-time traffic requirements. In the MAC layer, the *queuing module* maintains the data queues for each MSS and the *scheduling module* conducts the actions of the scheduler. In the PHY layer, the *channel-estimating module* simulates the channel conditions and estimates the transmission rate of each MSS, and the *burst-allocating module* conducts the actions of the burst allocator. The arrows in Fig. 6 show the interaction between all the modules in our simulator. In particular, the traffic-generating module will generate traffic and feed them to the scheduling module for allocating resources and to the queuing module for simulating the queue of each traffic. The channel-estimating module will send the transmission rates of MSSs to both the scheduling and burst allocating modules for their references. In addition, the scheduling and the burst-allocating modules will interact with each other, particularly for our scheme.

The simulator adopts a fast Fourier transform (FFT) size of 1024 and the zone category as PUSC with reuse 1. The frame duration is 5 ms. This way, we have $X = 12$ and $Y = 30$. Six MCSs are adopted, which are denoted by a set $MCS = \{\text{QPSK1/2}, \text{QPSK3/4}, \text{16QAM1/2}, \text{16QAM3/4}, \text{64QAM2/3}, \text{64QAM3/4}\}$. For the traffic-generating module, the types of real-time traffic include UGS, rtPS, and ertPS, whereas the types of non-real-time traffic include nrtPS and BE. Each MSS has an admitted real-time data rate R_i^{rt} of 0 ∼ 200 bits and an admitted non-real-time data rate R_i^{nrt} of 0 ∼ 500 bits/frame. In each frame, each MSS generates $0 \sim 2R_i^{rt}$ amount of real-time data and $R_i^{nrt} \sim 4R_i^{nrt}$ amount of non-real-time data.

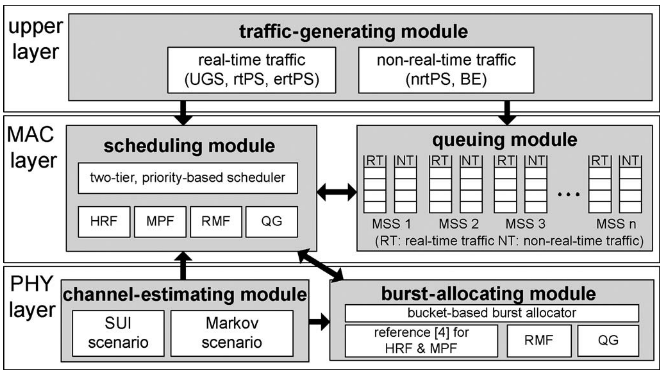

upper layer
traffic-generating module
real-time traffic (UGS, rtPS, ertPS)
non-real-time traffic (nrtPS, BE)
MAC layer
scheduling module
two-tier, priority-based scheduler
HRF
MPF
RMF
QG
queuing module
RT NT RT NT RT NT RT NT
MSS 1 MSS 2 MSS 3 MSS n
(RT: real-time traffic NT: non-real-time traffic)
PHY layer
channel-estimating module
SUI scenario
Markov scenario
burst-allocating module
bucket-based burst allocator
reference [4] for HRF & MPF
RMF
QG

Fig. 6. Architecture of our C++ simulator.

TABLE III
Amounts of Data Carried by Each Slot and the Minimum Required SNR Thresholds of Different MCSs

index	MCSs	data carried by each slot	minimum required SNR
1	QPSK 1/2	48 bits	6 dBm
2	QPSK 3/4	72 bits	8.5 dBm
3	16QAM 1/2	96 bits	11.5 dBm
4	16QAM 3/4	144 bits	15 dBm
5	64QAM 2/3	192 bits	19 dBm
6	64QAM 3/4	216 bits	21 dBm

For the channel-estimating module, we develop two scenarios to estimate the transmission rate of each MSS. The first scenario, called the *Stanford University Interim (SUI) scenario*, is based on the SUI path loss model recommended by the IEEE 802.16 Task Group [24]. In particular, each MSS will roam inside the BS's signal coverage (the largest area that the BS can communicate with each MSS using the lowest QPSK1/2 MCS) and move following the random waypoint model, with the maximal speed of 20 m/s [25]. The transmission rate of each MSS M_i is determined by its received SNR as

$$SNR(BS, M_i) = 10 \cdot \log_{10}\left(\frac{\tilde{P}(BS, M_i)}{BW \cdot N_o}\right)$$

where BW is the effective channel bandwidth (in hertz), N_o is the thermal noise level, and $\tilde{P}(BS, M_i)$ is the received signal power at M_i, which is defined by

$$\tilde{P}(BS, M_i) = \frac{G_{BS} \cdot G_{M_i} \cdot P_{BS}}{L(BS, M_i)}$$

where P_{BS} is the transmission power of the BS, G_{BS} and G_{M_i} are the antenna gains at the BS and M_i, respectively, and $L(BS, M_i)$ is the path loss from the BS to M_i. Given M_i's SNR, the BS can determine M_i's MCS based on Table III. In particular, the BS will choose the highest MCS whose minimum required SNR is smaller than $SNR(BS, M_i)$. Table IV lists the parameters used in the SUI scenario.

TABLE IV
SIMULATION PARAMETERS USED IN THE SUI SCENARIO

parameter	value
P_{BS}	1000 milliwatts
subchannel bandwidth (BW)	10 MHz
path loss model	SUI
antenna hight	BS: 30 meters; MSS: 2 meters
thermal noise	-100 dBm

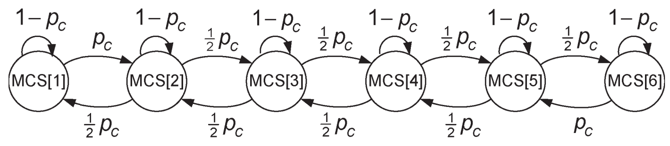


Fig. 7. Six-state Markov chain to model the channel condition.

The second scenario, called the Markov scenario, adopts a six-state Markov chain [26] to simulate the channel condition of each MSS, as shown in Fig. 7. In particular, let $MCS[i]$ be the ith MCS, $i = 1, \ldots, 6$. Suppose that an MSS uses $MCS[i]$ to fit its channel condition at the current frame. The probabilities that the MSS switches to $MCS[i-1]$ and $MCS[i+1]$ in the next frame are both $(1/2)p_c$, and the probability that it remains unchanged is $1 - p_c$. For the boundary cases of $i =$ 1 and 6, the probabilities of switching to $MCS[2]$ and $MCS[5]$, respectively, are both p_c. Unless otherwise stated, we set $p_c =$ 0.5, and the initial i value of each MSS is randomly selected from 2 to 5.

We compare our cross-layer framework with the *high-rate-first* (*HRF*) scheme [21], the *modified proportional fair* (*MPF*) scheme [10], the *rate maximization with fairness consideration* (*RMF*) scheme [20], and the *QoS guarantee* (*QG*) *scheme* [22]. HRF always first selects the MSS with the highest transmission rate C_i to serve. MPF assigns priorities to MSSs, where an MSS with a higher C_i value and a lower amount of received data is given a higher priority. RMF first allocates resources to unsatisfied MSSs according to their minimum requirements, where MSSs are sorted by their transmission rates. If there remain resources, they are allocated to the MSSs with higher transmission rates. Similarly, QG first satisfies the minimum requirements of each MSS's traffic, which are divided into real-time and non-real-time traffic. Then, the remaining resources are allocated to MSSs with higher transmission rates. Because both HRF and MPF implement only the scheduler, we adopt the scheme in [4] as their burst allocators. In our framework, we use $B = 5$ buckets and set $\gamma = 0.3$ in **P2**, unless otherwise stated. In Section VI-F, we will discuss the effects of these two parameters on the system performance. The duration of each experiment is at least 1000 frames.

A. Network Throughput

We first compare the network throughput under different number of MSSs (i.e., n), where the network throughput is defined by the amount of MSSs' data (in bits) transmitted by the BS during 1000 frames. We observe the case when the network becomes saturated, where there are 60 $\sim$ 90 MSSs to be served. Fig. 8 shows the simulation results under both the SUI and the Markov scenarios, where the trends are similar. Explicitly, when the number of MSSs grows, the throughput increases but will eventually become steady when there are too many MSSs (i.e., $n \geq 80$). The throughput under the SUI scenario is lower than the throughput under the Markov scenario, because some MSSs may move around the boundary of the BS's coverage, leading to a lower SNR and, thus, a lower MCS. Under the Markov scenario, a higher p_c means that each MSS may more frequently change its MCS, and vice versa.

In general, both RMF and QG ignore the effect of IE overheads on network performance so that their throughput will be degraded. Although HRF first serves MSSs with higher transmission rates, its throughput is not the highest. The reason is that HRF not only ignores the importance of IE overheads but also neglects the effect of C_i/C_i^{avg} factor on potential throughput when scheduling traffic. The throughput of MPF is higher than the throughput of RMF, QG, and HRF for the following two reasons. First, MPF prefers MSSs that use the higher transmission rates, which is similar to HRF. However, HRF incurs higher IE overheads because of the scheduling methodology (which will be verified in Section VI-B). Second, both RMF and QG try to schedule every traffic in each frame, which generates too many IEs (in fact, we can postpone scheduling some traffic to reduce IE overheads while still guaranteeing long-term fairness, as will be verified in Sections VI-B and C). On the other hand, MPF enjoys higher throughput, because it takes care of IE overheads from the viewpoint of the scheduler. In particular, our cross-layer framework has the highest throughput in most cases because of the following two reasons. First, our scheduler assigns a higher priority to MSSs with higher C_i and C_i/C_i^{avg} values and thus makes MSSs receive their data in higher transmission rates. Second, both our scheduler and burst allocator can effectively decrease the number of IEs and acquire more subframe space for data transmission. Note that, when $n = 90$, our cross-layer framework will try to satisfy a large number of urgent traffic to avoid their packets being dropped. In this case, its throughput is slightly lower than the throughput of MPF, but our cross-layer framework can significantly reduce the real-time packet-dropping ratio, as will be shown in Section VI-D.

B. IE Overheads and Subframe Utilization

Fig. 9 shows the average number of IEs in each downlink subframe. As discussed earlier, HRF, RMF, and QG do not consider IE overheads; therefore, they will generate a large number of IEs. The situation becomes worse when the number of MSSs grows, because each MSS needs to be allocated with at least one burst (and, thus, one IE). By considering IE overheads in the scheduler, MPF can reduce the average number of IEs per frame. It can be observed that, when the number of MSSs grows, the number of IEs in MPF reduces. The reason is that MPF allocates more resources to MSSs in a frame to reduce the total number of scheduled MSSs, thus reducing the number of allocated bursts (and IEs). In Fig. 9, our cross-layer framework generates the smallest number of IEs per frame, because both the proposed scheduler and burst allocator consider IE overheads, and the framework can adjust

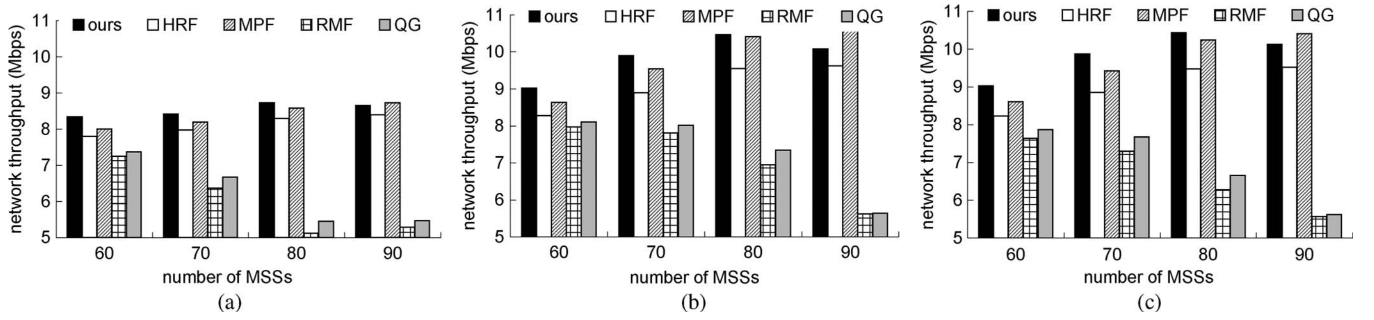


Fig. 8. Comparison on network throughput. (a) SUI scenario. (b) Markov scenario ($p_c = 0.8$). (c) Markov scenario ($p_c = 0.2$).

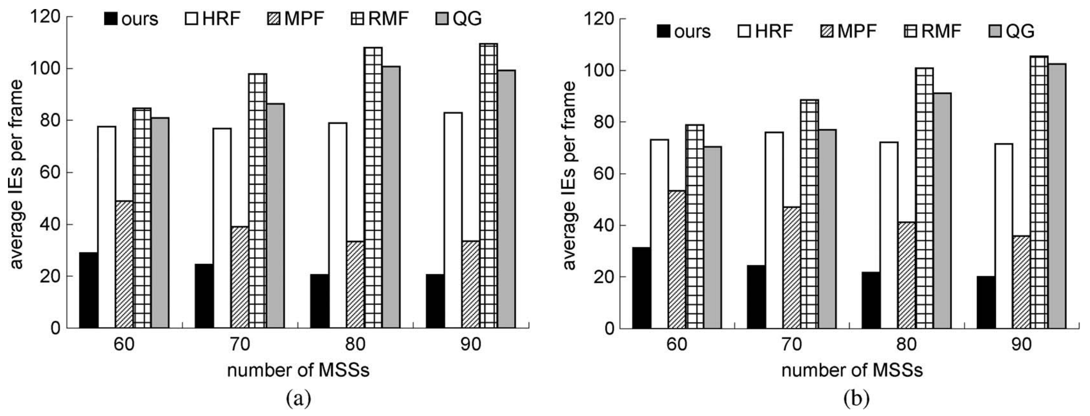


Fig. 9. Comparison on IE overheads. (a) SUI scenario. (b) Markov scenario.

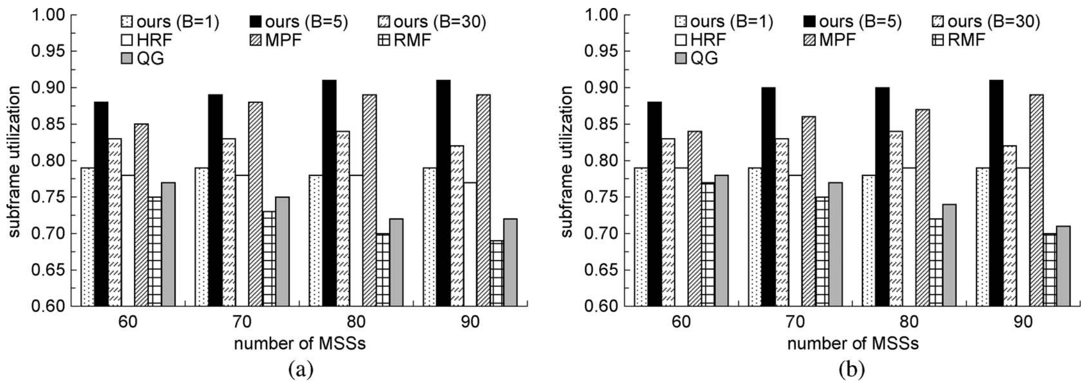


Fig. 10. Comparison on subframe utilization. (a) SUI scenario. (b) Markov scenario.

the number of nonurgent real-time traffic to be served to avoid generating too many bursts.

IE overheads have a strong impact on the utilization of downlink subframes, as reflected in Fig. 10. Because HRF, RMF, and QG generate a large number of IEs, their subframe utilization will be lower than MPF and our cross-layer framework. It can be observed that the number of buckets B significantly affects the subframe utilization of our cross-layer framework. In particular, a very large B (e.g., 30) will reduce the amount of data carried in each bucket and thus generate many small bursts. On the other hand, a very small B (e.g., 1) may degrade the functionality of buckets, and thus, some resource assignments may not fully utilize the bursts allocated to them. Based on Fig. 10, we suggest setting $B = 5$ to get the best utilization, and the analysis result in Section VI-G will also validate this point.

C. Long-Term Fairness

Next, we verify whether each scheme can guarantee long-term fairness under a highly congested network, where there are $140 \sim 200$ MSSs. Fig. 11 shows the fairness indices (FI) of all schemes. Recall that the network becomes saturated when there are 80 MSSs. Thus, it is impossible to get a FI of 1, because the network resource is not enough to satisfy the requirements of all traffic. Based on Fig. 11, HRF incurs the lowest index, because it always serves MSSs that use higher transmission rates. By considering the amount of allocated data of each MSS, MPF can have a higher index than HRF. QG and RMF try to satisfy the minimum requirement of each traffic in every frame, thus leading to higher indices. Because RMF allocates the resources to MSSs sorted by their transmission rates, its index will be lower than QG.

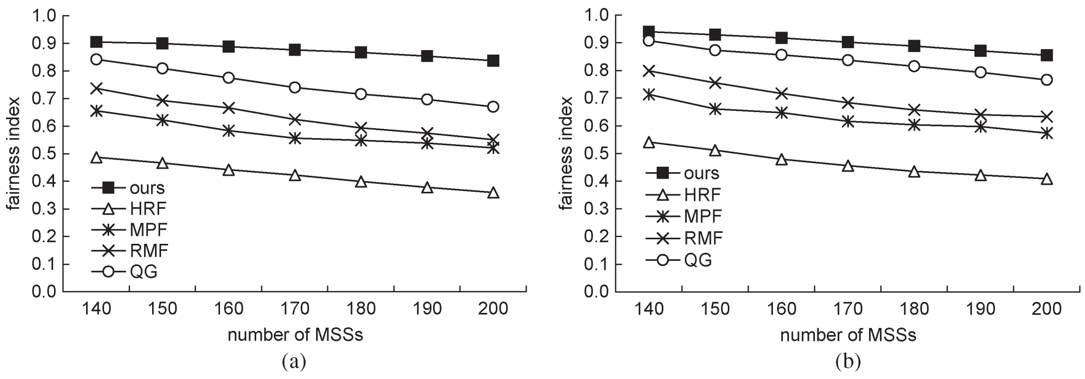


Fig. 11. Comparison on long-term fairness. (a) SUI scenario. (b) Markov scenario.

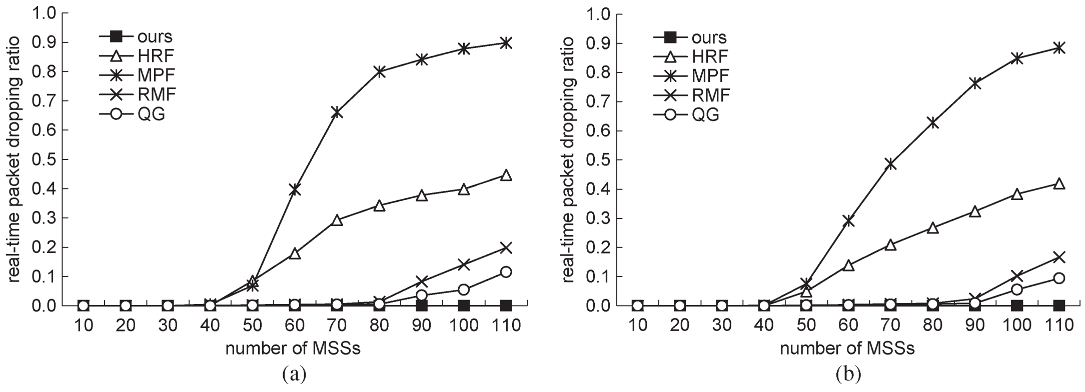


Fig. 12. Comparison on real-time packet-dropping ratios under different numbers of MSSs. (a) SUI scenario. (b) Markov scenario.

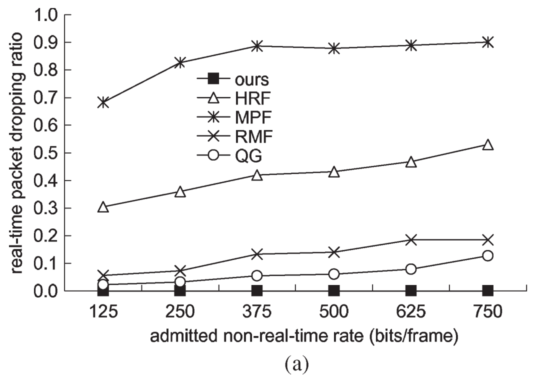
Our cross-layer framework has the highest FI (more than 0.85) due to the following two reasons. First, our priority-based scheduler only schedules γ ratio of nonurgent real-time traffic to avoid starving non-real-time traffic. Second, our cross-layer framework tries to reduce the IE overheads and acquire more frame space to allocate bursts for MSSs' traffic. In this case, we have more resources to fairly distribute among MSSs. Thus, our cross-layer framework can maintain long-term fairness, even in a highly congested network.

D. Packet-Dropping Ratios of Real-Time Traffic

We then observe the packet-dropping ratios of real-time traffic, where each MSS will generate $0 \sim 2R_i^{rt}$ real-time data in each frame. When a real-time packet is not transmitted within 6 frames (i.e., 30 ms) after being generated, it will be dropped. Fig. 12 shows the real-time packet-dropping ratios of all schemes under $10 \sim 110$ MSSs. Both HRF and MPF distribute resources to MSSs based on the transmission rates without considering the traffic types; therefore, their ratios begin raising when $n \geq 50$. In this case, a large amount of non-real-time traffic will compete with real-time traffic for the limited resource. On the other hand, the ratios of RMF and QG begin raising when $n \geq 90$. Because both RMF and QG try to satisfy the minimum requirements of all traffic in each frame, they can avoid real-time packet dropping when the network is not saturated (i.e., $n < 90$). Our cross-layer framework can have almost zero ratio due to the following three reasons. First, our priority-based scheduler assigns urgent real-time traffic with the highest priority. In addition, it schedules a γ ratio of nonurgent real-time traffic to avoid generating too many urgent traffic in the following frames. Second, our bucket-based burst allocator arranges bursts based on the priorities from the scheduler; therefore, the bursts of the urgent real-time traffic can first be allocated to avoid packet dropping. Third, both our scheduler and burst allocator try to reduce IE overheads, and thus, more urgent real-time traffic can be served in each frame.

Fig. 13 shows the real-time packet-dropping ratios of all schemes under different admitted non-real-time data rates, where the network is saturated. Because MPF proportionally distributes resources among MSSs, it incurs the highest real-time packet-dropping ratio. On the other hand, because some MSSs with real-time traffic may have higher transmission rates, the ratio of HRF is lower than the ratio of MPF. As discussed earlier, both RMF and QG try to satisfy the minimum requirement of each traffic, and thus, their ratios can become lower. Note that, because QG differentiates real-time traffic from non-real-time traffic, its ratio is lower than the ratio of RMF. Our cross-layer framework always has a zero ratio, because the bursts of urgent real-time traffic are first allocated, and our framework can acquire more frame space to serve urgent real-time traffic by reducing IE overheads.

Because the trends under both SUI and Markov scenarios are similar, we only show the results under the Markov scenario in the following experiments.



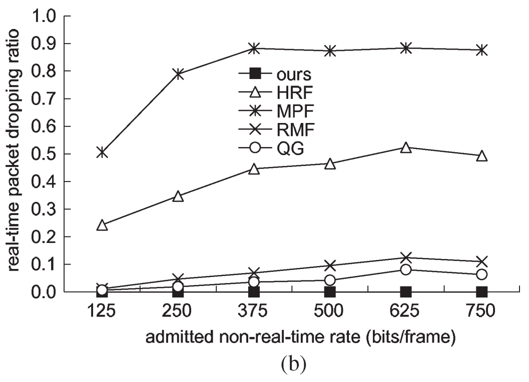


Fig. 13. Comparison on real-time packet-dropping ratios under different admitted non-real-time rates. (a) SUI scenario. (b) Markov scenario.

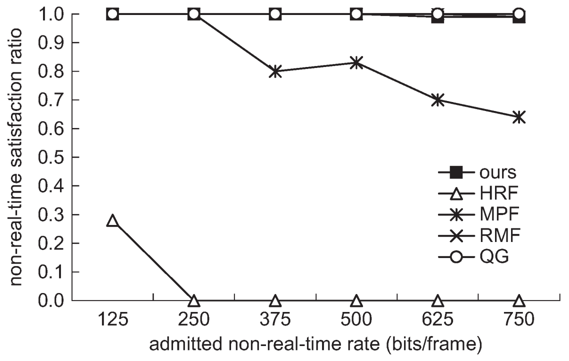


Fig. 14. Comparison on non-real-time satisfaction ratios of the bottom 10% MSSs under the Markov scenario.

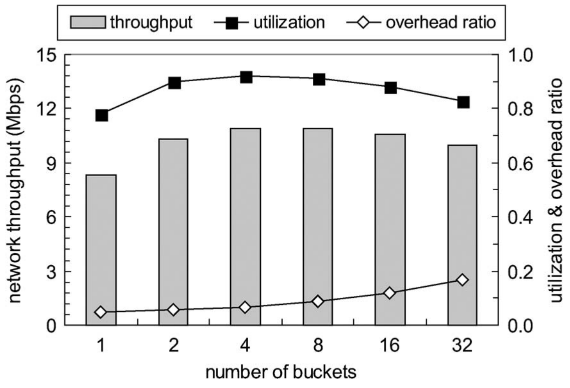


Fig. 15. Effect of the number of buckets B on the network throughput, subframe utilization, and IE overheads under the Markov scenario.

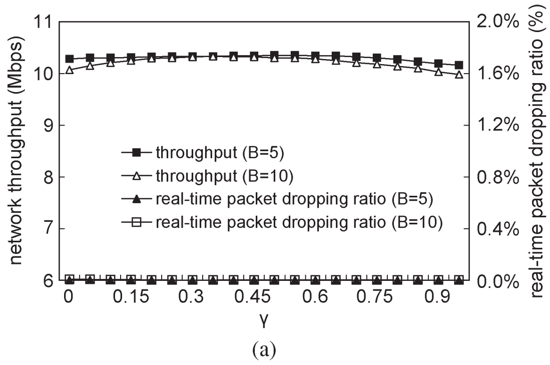
E. Satisfaction Ratios of Non-Real-Time Traffic

Next, we measure the satisfaction ratios of non-real-time traffic [by (5)] under a saturated network. Fig. 14 shows the satisfaction ratios of non-real-time traffic of the bottom 10% MSSs. When the non-real-time rate is larger than 125 bits/frame, the ratio of HRF is zero, because these bottom 10% MSSs (whose transmission rates must be lower) are starved. The ratio of MPF starts diminishing when the non-real-time rate is larger than 250 bits/frame, because MPF proportionally distributes resources among traffic. By satisfying the minimum requirement of each traffic, the ratios of RMF and QG are close to one. Our cross-layer framework can have a ratio of nearly one for the bottom 10% MSSs, which means that non-real-time traffic will not be starved, although our scheme prefers real-time traffic.

F. Effects of System Parameters

We then observe the effects of system parameters in our cross-layer framework on network throughput, subframe utilization, and IE overheads under a saturated network (i.e., the number of MSS is 90). Fig. 15 shows the impact of the number of buckets (i.e., B) on network throughput, utilization, and overhead ratios when $Y = 32$. Here, the *overhead ratio* is defined as the ratio of the number of slots used for MAP information (e.g., DL-MAP, UL-MAP, and IEs) to the total number of slots in a downlink subframe. In general, the utilization decreases when the overhead ratio increases, because they are complementary. Based on Fig. 15, the utilization first increases and then decreases when B grows. The former increment is because some resource assignments do not fully utilize their allocated bursts. On the other hand, the latter decrement is because the burst allocator generates too many bursts to satisfy the thinner buckets. The overhead ratio increases when B increases, because more IEs are generated. In addition, when $B \leq 4$, the throughput increases when B grows, because more buckets may serve more requests. On the other hand, when $B \geq 8$, such a trend reverses, because more IEs are generated, causing lower utilization. Based on Fig. 15, we suggest setting $B = 4 \sim 8$, because this range of B value improves both throughput and utilization while reducing IE overheads.

Fig. 16 shows the effects of γ and B on real-time packet-dropping ratios and network throughput in our cross-layer framework. Explicitly, the real-time packet-dropping ratio decreases when γ grows, because more real-time traffic can be served. However, when γ increases, the throughput may decrease, because the scheduler has to select more nonurgent real-time traffic to serve. In this case, some real-time traffic with lower transmission rates may be served, which degrades the throughput. As aforementioned, a large B may generate more IEs and thus reduce the utilization. Thus, the throughput in the case of larger B (e.g., $B = 15$ and 30) starts dropping earlier than in the case of smaller B (e.g., $B = 5$ and 10). Based on Fig. 16, we suggest setting $\gamma = 0.15 \sim 0.45$, because this range of γ value not only improves the network throughput but also reduces real-time packet-dropping ratios under different values of B.



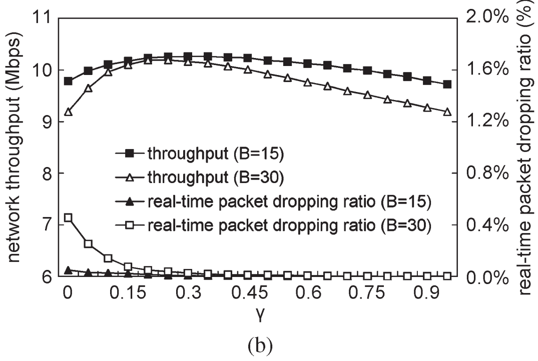


Fig. 16. Effect of γ on the network throughput and real-time packet-dropping ratios under the Markov scenario. (a) Smaller B. (b) Larger B.

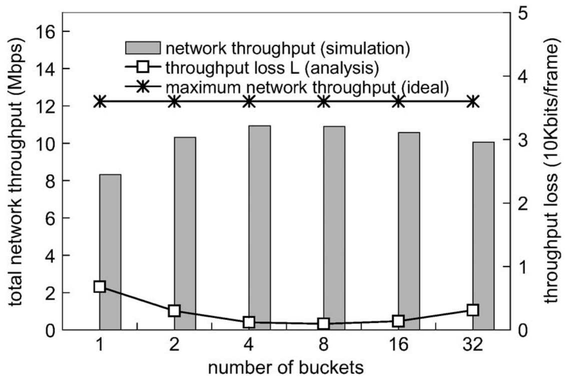


Fig. 17. Effect of the number of buckets (B) on the throughput loss $\mathcal{L}$ (by analysis) and the total network throughput (by simulation) under the Markov scenario.

G. Verification of Throughput Analysis

Finally, we verify our analytic results in the part where two transmission rates, $c_{\text{low}} = 48$ bits/slot and $c_{\text{high}} = 96$ bits/slot, are adopted. The probabilities that an MSS can use c_{low} and c_{high} are both 0.5. Then, the probability that m MSSs can use c_{low} is

$$\begin{aligned} Prob[\tilde{N}_L = m] &= C_m^n \times (Prob[\text{transmission rate} = c_{\text{low}}])^m \\ &\quad \times (Prob[\text{transmission rate} = c_{\text{low}}])^{n-m} \\ &= \frac{n!}{m!(n-m)!} \times (0.5)^m \times (0.5)^{n-m} \\ &= \frac{(0.5)^n \times n!}{m!(n-m)!}. \end{aligned}$$

In addition, the probability that an MSS M_i has urgent data is

$$Prob\left[I_i^U = 1\right] = (1-\gamma)^{T_D - 1}$$

where T_D is the deadline of real-time data that will be dropped (in frames). Note that, because the scheduler will serve all queued real-time data from the top γn MSSs in each frame, after $(T_D - 1)$ frames, the probability of an MSS with urgent data is no more than $(1-\gamma)^{T_D-1}$. In our simulation, we set $\gamma = 0.3$, $T_D = 6$ frames, and $\mathcal{R} = 200$ bits/frame.

Fig. 17 shows the analysis and simulation results. When $B < 4$, the throughput loss $\mathcal{L}$ decreases, but the network throughput increases as B increases. On the other hand, when $B > 8$, $\mathcal{L}$ increases, but the network throughput decreases as B increases. This result indicates that the minimum value of $\mathcal{L}$ by analysis appears at the range of $B = [4, 8]$, whereas the maximum network throughput by simulation appears at the same range of $B = [4, 8]$. Thus, our analysis and simulation results are consistent. Based on Fig. 17, we suggest setting $B = 4 \sim 8$ to maximize the network throughput and minimize $\mathcal{L}$, which matches the results in Fig. 15. Therefore, our analysis can validate the simulation results and provide guidelines for the setting of the burst allocator.

VII. Conclusion

In this paper, we have proposed a cross-layer framework that covers the issues of overhead reduction, real-time and non-real-time traffic scheduling, and burst allocation in an IEEE 802.16 OFDMA network. Compared with existing solutions, our framework is more complete, because it involves codesigning both the two-tier priority-based scheduler and the bucket-based burst allocator. Our scheduler reduces potential IE overheads by adjusting the number of MSSs to be served. With a two-tier priority rules, it guarantees real-time traffic delays, ensures satisfaction ratios of non-real-time traffic, and maintains long-term fairness. On the other hand, our burst allocator incurs low complexity and guarantees a bounded number $(k + (Y/\Delta_{bkt}) - 1)$ of IEs to accommodate data bursts. In addition, it follows the priority rule from the scheduler to avoid packet dropping of urgent real-time traffic. We have also analyzed the impact of the number of buckets on the throughput loss. Through both analyses and simulations, we show how we can adjust the system parameters to reduce IE overheads, improve subframe utilization, and enhance network throughput. In addition, these results verify that such a cross-layer framework significantly improves the resource allocation and utilization of downlink communications in WiMAX networks. For future work, we will investigate how we can optimize the scheduler and burst allocator for some particular cases, e.g., various traffic characteristics and MSS densities. In addition, we will consider extending our results for WiMAX relay networks.

References

[1] *IEEE Standard for Local and Metropolitan Area Networks Part 16—Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands and Corrigendum 1*, IEEE Std. 802.16e-2005, 2006.

[2] B. Rong, Y. Qian, and H. H. Chen, "Adaptive power allocation and call admission control in multiservice WiMAX access networks," *IEEE Wireless Commun.*, vol. 14, no. 1, pp. 14–19, Feb. 2007.

[3] K. Sundaresan and S. Rangarajan, "Efficient algorithms for leveraging spatial reuse in OFDMA relay networks," in *Proc. IEEE INFOCOM*, 2009, pp. 1539–1547.

[4] Y. Ben-Shimol, I. Kitroser, and Y. Dinitz, "Two-dimensional mapping for wireless OFDMA systems," *IEEE Trans. Broadcast.*, vol. 52, no. 3, pp. 388–396, Sep. 2006.

[5] H. S. Kim and S. Yang, "Tiny MAP: An efficient MAP in IEEE 802.16/WiMAX broadband wireless access systems," *Comput. Commun.*, vol. 30, no. 9, pp. 2122–2128, Jun. 2007.

[6] Y. Ma and D. Kim, "Rate-maximization scheduling schemes for uplink OFDMA," *IEEE Trans. Wireless Commun.*, vol. 8, no. 6, pp. 3193–3205, Jun. 2009.

[7] J. Shi and A. Hu, "Maximum utility-based resource allocation algorithm in the IEEE 802.16 OFDMA system," in *Proc. IEEE ICC*, 2008, pp. 311–316.

[8] R. Pitic and A. Capone, "An opportunistic scheduling scheme with minimum data-rate guarantees for OFDMA," in *Proc. IEEE WCNC*, 2008, pp. 1716–1721.

[9] N. A. Ali, M. Hayajneh, and H. Hassanein, "Cross-layer scheduling algorithm for IEEE 802.16 broadband wireless networks," in *Proc. IEEE ICC*, 2008, pp. 3858–3862.

[10] J. Kim, E. Kim, and K. S. Kim, "A new efficient BS scheduler and scheduling algorithm in WiBro systems," in *Proc. IEEE ICACT*, 2006, vol. 3, pp. 1467–1470.

[11] T. Wang, H. Feng, and B. Hu, "Two-dimensional resource allocation for OFDMA system," in *Proc. IEEE Int. Conf. Commun. Workshops*, 2008, pp. 1–5.

[12] T. Ohseki, M. Morita, and T. Inoue, "Burst construction and packet-mapping scheme for OFDMA downlinks in IEEE 802.16 systems," in *Proc. IEEE GLOBECOM*, 2007, pp. 4307–4311.

[13] X. Perez-Costa, P. Favaro, A. Zubow, D. Camps, and J. Arauz, "On the challenges for the maximization of radio resources usage in WiMAX networks," in *Proc. IEEE CCNC*, 2008, pp. 890–896.

[14] A. Erta, C. Cicconetti, and L. Lenzini, "A downlink data region allocation algorithm for IEEE 802.16e OFDMA," in *Proc. IEEE Int. Conf. Inf., Commun. Signal Process.*, 2007, pp. 1–5.

[15] T. Kwon, H. Lee, S. Choi, J. Kim, D. H. Cho, S. Cho, S. Yun, W. H. Park, and K. Kim, "Design and implementation of simulator based on cross-layer protocol between MAC and PHY layers in WiBro compatible IEEE 802.16e OFDMA system," *IEEE Commun. Mag.*, vol. 43, no. 12, pp. 136–146, Dec. 2005.

[16] Intel ships its next-generation WiMAX chip with support for mobile networks. [Online]. Available: http://www.intel.com/pressroom/archive/releases/2006/20061011comp.htm

[17] Y. Tcha, M. S. Kim, and S. C. Lee, "A compact MAP message to provide a virtual multiframe structure for a periodic fixed-bandwidth assignment scheme," IEEE Std. C802.16e-04/368r2, 2004.

[18] J. Kim and D. H. Cho, "Piggybacking scheme of MAP IE for minimizing MAC overhead in the IEEE 802.16e OFDMA systems," in *Proc. IEEE VTC*, 2007, pp. 284–288.

[19] K. Kim, Y. Han, and S. L. Kim, "Joint subcarrier and power allocation in uplink OFDMA systems," *IEEE Commun. Lett.*, vol. 9, no. 6, pp. 526–528, Jun. 2005.

[20] L. Gao and S. Cui, "Efficient subcarrier, power, and rate allocation with fairness consideration for OFDMA uplink," *IEEE Trans. Wireless Commun.*, vol. 7, no. 5, pp. 1507–1511, May 2008.

[21] X. Zhu, J. Huo, S. Zhao, Z. Zeng, and W. Ding, "An adaptive resource allocation scheme in OFDMA-based multiservice WiMAX systems," in *Proc. IEEE ICACT*, 2008, pp. 593–597.

[22] X. Zhu, J. Huo, X. Xu, C. Xu, and W. Ding, "QoS-guaranteed scheduling and resource allocation algorithm for IEEE 802.16 OFDMA system," in *Proc. IEEE ICC*, 2008, pp. 3463–3468.

[23] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *J. Comput. Netw. ISDN*, vol. 17, no. 1, pp. 1–14, Jun. 1989.

[24] Channel models for fixed wireless applications. [Online]. Available: http://www.ieee802.org/16/tga/docs/80216a-03_01.pdf

[25] C. Bettstetter, G. Resta, and P. Santi, "The node distribution of the random waypoint mobility model for wireless ad hoc networks," *IEEE Trans. Mobile Comput.*, vol. 2, no. 3, pp. 257–269, Jul.–Sep. 2003.

[26] H. S. Wang and N. Moayeri, "Finite-state Markov channel—A useful model for radio communication channels," *IEEE Trans. Veh. Technol.*, vol. 44, no. 1, pp. 163–171, Feb. 1995.