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

Jia-Ming Liang, Jen-Jee Chen, You-Chiun Wang, and Yu-Chee Tseng

Abstract—IEEE 802.16 OFDMA downlink subframes have a special 2D channel-time structure. Allocation resources from such a 2D structure incurs extra control overheads that hurt network performance. Existing solutions try to improve network performance by designing solely either the scheduler in the MAC layer or the burst allocator in the physical layer, but the efficiency of overhead reduction is limited. In the paper, we point out the necessity of 'co-designing' both the scheduler and the burst allocator to efficiently reduce overheads and improve network performance. Under the PUSC 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 traffics. Given available space and bucket design from the burst allocator, the scheduler can well utilize 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, WiMAX OFDMA

1 Introduction

The IEEE 802.16 standard [1] is developed for wide-range broadband wireless access. The physical layer employs the OFDMA (orthogonal frequency division multiple access) technique, where a *base station (BS)* can communicate with multiple *mobile subscriber stations (MSSs)* simultaneously through a set of orthogonal subchannels. The standard supports the FDD (frequency division duplex) and the TDD (time division duplex) modes; this paper aims at the TDD mode. Under the TDD mode, two types of subcarrier grouping models are specified: *AMC (adaptive modulation and coding)* and *PUSC (partial usage of subcarriers)*. 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) since the BS calculates the average quality for each subchannel based on MSSs' reports [2], [3].

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, wangyc, 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

The BS manages network resources for MSSs' data traffics, which are classified into *real-time traffics* (*e.g.*, unsolicited grant service (UGS), real-time polling service (rtPS), and extended rtPS (ertPS)) and *non-real-time traffics* (*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 2D 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 2D sub-array and needs to be specified by a *downlink map information element* (*DL-MAP_IE* or simply *IE*) in the DL-MAP field. These IEs are encoded by the robust QPSK1/2 modulation and coding scheme for reliability. Because the IEs occupy frame space and do not carry MSSs' data, they are considered as *control overheads*. Explicitly, how to efficiently reduce IE overheads will significantly affect network performance since it determines frame utilization. To manage resources to all data traffics, the standard defines a *scheduler* in the MAC layer and a *burst allocator* in the physical layer. However, their designs are left as open issues to implementers.

This paper aims at *co-designing* 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 three issues:

- The scheduler should improve network throughput while maintain long-term fairness. Since the BS may send data to MSSs using different transmission rates (due to network situations), the scheduler will prefer those MSSs using higher transmission rates but should avoid starving those MSSs using lower transmission rates.

- The scheduler should satisfy the delay constraints of real-time traffics to avoid high packet dropping ratios. However, it should also meet the requirements of non-real-time traffics.
- To well utilize the limited frame space, the scheduler has to reduce IE overheads when assigning resources to MSSs' data traffics. This 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 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, since all subchannels are equally adequate for all MSSs, the problem of arranging IEs and downlink bursts will become a 2D 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 requires a co-designing 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, those bursts for urgent real-time traffics should be allocated first 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 just take care of either one of them, 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 may cause unfairness among MSSs and high packet dropping ratios of real-time traffics. Therefore, it is necessary to co-design 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' traffics in a two-tier manner and allocates resources to these traffics based on their priorities. In the first tier, traffics are differentiated by their types. Urgent real-time traffics are assigned with the highest level-1 priority to avoid their packets being dropped in the next frame. Then, a γ ratio ($0 < \gamma < 1$) of non-urgent real-time traffics are assigned with level-2 priority and non-real-time traffics are given with level-3 priority. The above design has two advantages. First, we can avoid generating too many urgent real-time traffics in the next frame. Second, non-real-time traffics can have opportunity to be served to avoid being starved. In the second tier, traffics with the same type (*i.e.*, the same priority level in the first tier) are assigned with different priorities calculated by their 1) current transmission rates, 2) average transmission rates, 3) admitted data rates, and 4) queue lengths. The BS can have the knowledge of the above four factors because all downlink traffics are 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 traffics [10], our novel two-tier priority scheduling scheme not only prevents urgent real-time traffics from incurring packet dropping (through the first tier) but also maintains long-term fairness (through the second tier). The network throughput is also improved by giving a higher priority to those MSSs using higher transmission rates (in the second tier). In addition, the scheduler can adjust the number of MSSs to be served and assign resources to traffics according to the burst arrangement manner (from the burst allocator) to significantly reduce IE overheads. This design is neglected in prior studies and has 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 which 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 above bucket-based design incurs very low computation complexity and can be implemented on most low-cost 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 traffics scheduling, and burst allocation.

Major contributions of this paper are four-fold. First, we point out the necessity of co-designing 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 2 surveys related work. Section 3 gives the problem formulation.

The cross-layer framework is proposed in Section 4. Section 5 analyzes the expected network throughput of the proposed framework. Extensive simulation results are given in Section 6. Conclusions are drawn in Section 7.

2 Related Work

Most of prior studies on resource allocation in 802.16 OFDMA networks solely implement either the scheduler or the burst allocator. For the implementation of the scheduler, the study of [6] proposes a scheduling scheme according to MSSs' signal-to-noise ratios to achieve rate maximization. The work of [7] proposes a utility function to evaluate the tradeoff between network throughput and long-term fairness. In the work of [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 traffics. The work of [9] tries to minimize the blocking probability of MSSs' traffic requests and thus the packet dropping ratios of real-time traffics may be reduced. Nevertheless, all of the above studies [6]–[9] do not address the issue of overhead reduction. The work of [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, not only the efficiency of overhead reduction becomes insignificant but also some important data (*e.g.*, urgent real-time traffics) 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 of [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 in modifying the standard. The work of [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 pre-segments 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 of [11] allocates bursts for large requests first. Nevertheless, larger requests may not be necessarily more important or urgent. Several studies consider allocating bursts in a column-by-column manner. In the work of [12], bursts with the same modulation and coding scheme 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 of [13] pads zero bits in each column's end, which may cause low subframe utilization. The work of [14] adopts a backward, column-wise 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 can be seen, 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 IE overhead. The studies of [19], [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 traffics. The studies of [21], [22] develop an one-tier priority-based scheduler to allocate resources to each MSS to exactly satisfy its demand. Thus, the delay requirement of real-time traffics could be guaranteed but 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 6 that, without reducing IE overhead, the QoS (quality of service) requirements of MSSs' traffics may not be satisfied, especially when the network becomes saturated.

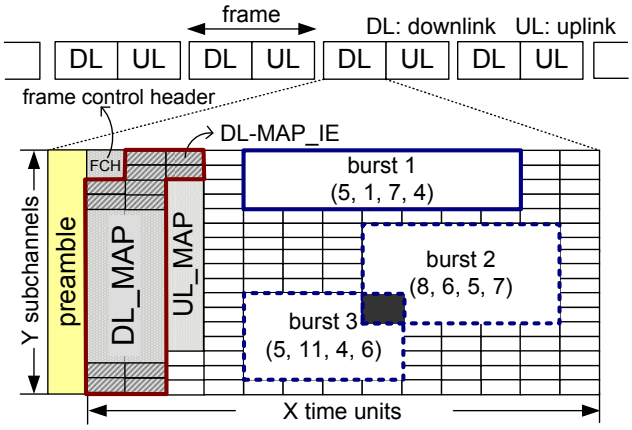


Fig. 1: The structure of an IEEE 802.16 OFDMA downlink subframe under the TDD mode.

Table 1 compares the features of prior studies and our cross-layer framework. It can be shown that our cross-layer framework 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 of WiMAX low-cost chips.

3 Resource Allocation Problem

We consider the downlink communication in an 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 1). 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 (referring to Fig. 1). A downlink subframe is modeled by a 2D 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 *slot*). Each downlink subframe is composed of three portions: *preamble*, *control*, and *data*. The control portion contains a DL-MAP and a 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 *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

TABLE 1: 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
references [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.

examples. Bursts 1 and 2 can coexist, but bursts 2 and 3 cannot coexist. Each burst requires one IE in 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 bits encoded by the QPSK1/2 modulation and coding scheme [5]. Since each slot can carry 48 bits by QPSK1/2, an IE occupies $\frac{5}{4}$ slots, which has 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, $i = 1..n$, 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 containing both the scheduler and the burst allocator to arrange bursts to MSSs, such that we can reduce IE overhead, improve 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 too complicated so that it can execute within a frame duration (*i.e.*, 5 ms) and 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:

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

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

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

where $\tilde{A}_i^{rt}(x)$ and $\tilde{A}_i^{nrt}(x)$ are the amounts of real-time and non-real-time traffics 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 $U_d(x)$ 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 $\frac{\sum_{j=0}^{T-1} U_d(f_c - j)}{T}$. Table 2 summarizes the notations used in this paper.

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 to conduct the estimation in Section 6.

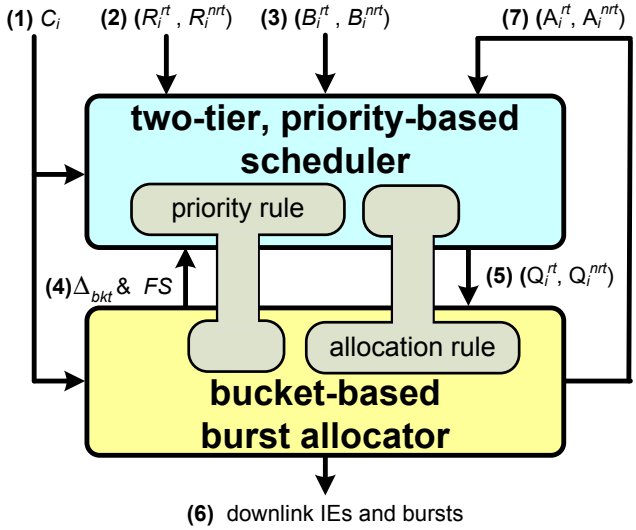


Fig. 2: The system architecture of the proposed cross-layer framework, where $i = 1..n$.

4 The Proposed Cross-Layer Framework

Fig. 2 shows the system architecture of our cross-layer framework, which is composed of two components: the *two-tier, priority-based scheduler* and the *bucket-based burst allocator*. The transmission rate C_i for each MSS M_i (label 1 in Fig. 2) is periodically reported to the scheduler and the burst allocator. Each M_i's admitted rates R_i^{rt} and R_i^{nrt} (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} (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 (label 4) to help the scheduler distribute resources among MSSs' traffics, 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 since the uplink subframe is allocated before the downlink subframe. The DL-MAP control fields contain all parts of DL-MAP except 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} (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 (label 6). The actual real-time and non-real-time traffics allocated to M_i are written as A_i^{rt} and A_i^{nrt} (label 7) and are fed to the scheduler for future scheduling.

TABLE 2: 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 traffics
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

In our cross-layer framework, the *priority rule* defined in the scheduler helps the burst allocator to determine how to arrange bursts for MSSs' traffics. On the other hand, the *allocation rule* defined in the burst allocator also helps the scheduler to determine how to assign resources to MSSs' traffics. Both the priority and allocation rules are like 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 to arrange 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 sliced horizontally into a number of buckets, each of size Δ_{bkt} (see Fig. 3 for an example). The size Δ_{bkt} actually serves as the allocation unit in our scheme. As to be seen, the scheduler always keeps $(Q_i^{rt} + Q_i^{nrt})$ as a multiple of Δ_{bkt} for each M_i. In this way, the burst allocator can easily arrange bursts in a 'bucket-by-bucket' manner, well utilize frame resource, and generate quite few bursts and thus IEs (which will be proved having an upper bound later in Section 4.2). 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 $i = 1..n$.

4.1 Two-Tier, Priority-Based Scheduler

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

- **P1**. Urgent real-time traffics whose packets will pass their deadlines at the end of this frame.
- **P2**. Real-time traffics ranked top γ ratio ($0 < \gamma < 1$) sorted by their importance.
- **P3**. Non-real-time traffics sorted by their importance.

Then, in the second tier, traffics with the same type are assigned with different priorities by their *importance*, which is calculated by their 1) current transmission rates, 2) average transmission rates, 3) admitted data rates, and 4) queue lengths. In particular, for priority level **P2**, we rank the *importance* of M_i's real-time traffic by

$$I_i^{rt} = C_i \times \frac{C_i}{C_i^{avg}} \times \frac{B_i^{rt}}{R_i^{rt}}. \quad (3)$$

Here, the importance I_i^{rt} involves three factors multiplied together:

1) A higher transmission rate C_i gives M_i a higher rating to improve network throughput.
2) A higher ratio $\frac{C_i}{C_i^{avg}}$ 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. Specifically, 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.*, $C_i > C_i^{avg}$). In addition, a higher $\frac{C_i}{C_i^{avg}}$ value means that the MSSs is currently in a better condition so that we give it a higher priority to improve the potential throughput.
3) A higher ratio $\frac{B_i^{rt}}{R_i^{rt}}$ 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

$$I_i^{nrt} = C_i \times \frac{C_i}{C_i^{avg}} \times \frac{1}{S_i^{nrt}}, \quad (4)$$

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

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

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

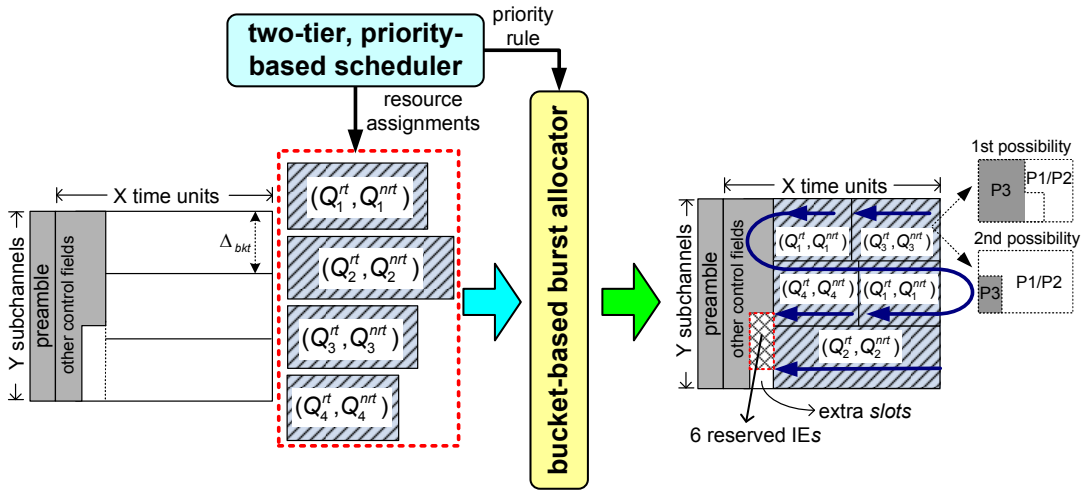


Fig. 3: An example of the bucket-based burst allocation with three buckets and four resource assignments.

Below, 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, $i = 1..n$. Initially, we set all $e_i = 0, i = 1..n$. Besides, the free space FS is deducted by $(\frac{Y}{\Delta_{bkt}} - 1) \times \theta_{IE}$ to preserve the space for potential IEs caused by the burst allocator (this will be discussed in the next section), where $\theta_{IE} = \frac{5}{4}$ is the size of an IE.

1) Let U_i^{rt} be the data amount 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 $FS = FS - \lceil \frac{Q_i^{rt}}{C_i} \rceil$. 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 traffics according to their I_i^{rt} values (by Eq. (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 resource $\delta = \min\{FS \times C_i, B_i^{rt}\}$ to M_i. Then, set $Q_i^{rt} = Q_i^{rt} + \delta$ and $FS = FS - \lceil \frac{\delta}{C_i} \rceil$. 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 Eq. (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 resource $\delta = \min\{FS \times C_i, B_i^{nrt}\}$ to M_i. Then, set $Q_i^{nrt} = \delta$ and $FS = FS - \lceil \frac{\delta}{C_i} \rceil$. Deduct δ from B_i^{nrt}.
4) Since 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 $(\frac{Q_i^{rt}+Q_i^{nrt}}{C_i} \mod \Delta_{bkt}) = 0$. One exception is when most of 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 $(\frac{Q_i^{rt}+Q_i^{nrt}}{C_i} \mod \Delta_{bkt}) = 0$. The above 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 important 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.

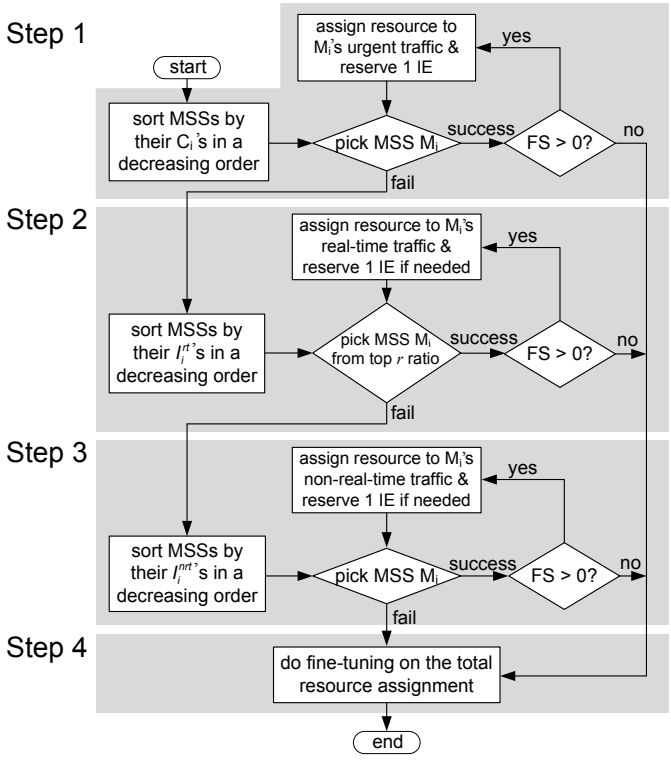


Fig. 4: The flowchart of the two-tier, priority-based scheduler.

Fig. 4 illustrates the flowchart of the scheduler. To summarize, our scheduler generates the resource assignment according to three priorities: (**P1**) urgent traffics, (**P2**) real-time traffics, and (**P3**) non-real-time traffics. Step 1 first schedules those MSSs with urgent traffics to alleviate their real-time traffic delays. Step 2 schedules those top γ ratio of MSSs to reduce the number of MSSs that may have urgent traffics in the following frames. This step also helps reduce the IE overhead of future frames caused by urgent traffics, which is neglected by prior studies. Step 3 schedules those 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 to 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 traffics 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 traffics takes $O(n)$ time. In step 4, reshaping all requests spends at most $O(n)$ time. Thus, the total time complexity is $O(n \lg n + n + n \lg n + \gamma n + n \lg n + n + n) = O(n \lg n)$.

4.2 Bucket-Based Burst Allocator

Ideally, the free space FS in Eq. (2) should accommodate each resource assignment (Q_i^{rt}, Q_i^{nrt}) calculated by the scheduler and its corresponding IE(s). However, since 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 originally arranged by the scheduler may be trimmed down or even kicked out by the burst allocator. Given the free space FS by Eq. (2), bucket size Δ_{bkt}, and assignments (Q_i^{rt}, Q_i^{nrt})'s from the scheduler, our bucket-based heuristic works as follows:

1) Slice FS horizontally[2] into $\frac{Y}{\Delta_{bkt}}$ 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 $\lceil (k + \frac{Y}{\Delta_{bkt}} - 1) \times \theta_{IE} \rceil$ 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. Since 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. Since each assignment (Q_i^{rt}, Q_i^{nrt}) may have data mixed in 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 slice FS vertically, but the effect will be the same.

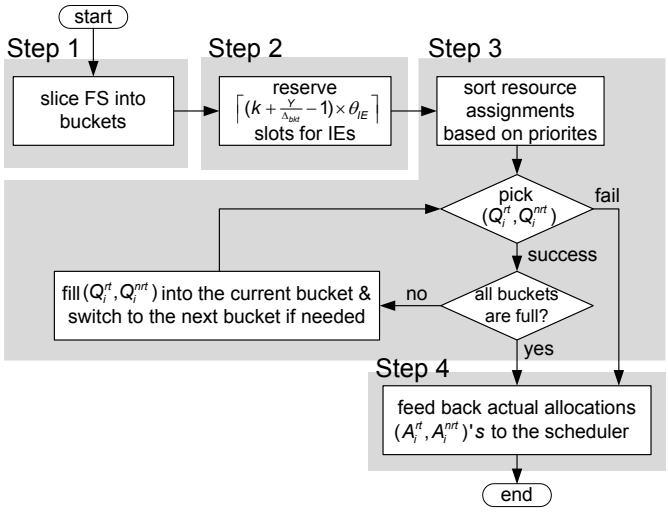


Fig. 5: The flowchart of the bucket-based burst allocator.

Then, bursts are allocated in a bucket-by-bucket manner. Specifically, 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 later case, we will move to the right end of the next bucket and repeat the above allocation process again. In addition, this 'cross-bucket' behavior will require one extra IE for the request. The above operation is repeated until either all assignments are examined or all buckets are exhausted. Fig. 3 gives an 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 since it involves in 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). Besides, 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 below. First, because there are $\frac{Y}{\Delta_{bkt}}$ buckets, there are at most $\left(\frac{Y}{\Delta_{bkt}} - 1\right)$ *cross-bucket* burst assignments and thus at most $\left(\frac{Y}{\Delta_{bkt}} - 1\right)$ extra IEs are needed. To accommodate this need, some assignments may be trimmed down slightly. This is why (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 which 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 above process for ease of presentation, but they can be used to allocate bursts to further improve space efficiency. Third, since each cross-bucket behavior will require one extra IE and there are $\frac{Y}{\Delta_{bkt}}$ buckets, the number of IEs required is bounded, as proved in Theorem 1.

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

Proof: Given $\frac{Y}{\Delta_{bkt}}$ buckets $\hat{b}_1, \hat{b}_2, \cdots$, and $\hat{b}_{\frac{Y}{\Delta_{bkt}}}$, we can concatenate them into one virtual bucket $\hat{b}$ with $\left(\frac{Y}{\Delta_{bkt}} - 1\right)$ joints. We then allocate one virtual burst for each request from the scheduler in $\hat{b}$, so 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 $\left(\frac{Y}{\Delta_{bkt}} - 1\right)$ joints is crossed by one virtual burst. In this case, we require $\left(k + \frac{Y}{\Delta_{bkt}} - 1\right)$ real bursts to replace all virtual bursts. Since each real burst requires one IE, we have to reserve at most $\left(k + \frac{Y}{\Delta_{bkt}} - 1\right)$ 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 be dynamically adjusted to reflect the 'grain size' of our allocation. A larger grain size may cause fewer IEs, but sacrifice resource utilization; a smaller grain size may cause more IEs, but improve resource utilization. We will discuss the effect of Δ_{bkt} in Section 6.6.

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

To conclude, the proposed scheduler and burst allocator are dependent with 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' traffics 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 traffics, urgent real-time traffics can be still arranged with bursts to catch their approaching deadlines.

5 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 traffics of priority levels **P1** and **P3**, and each MSS has infinite data in **P3**. (Traffics of **P2** will eventually become urgent traffics of **P1**.) Then, we calculate the difference between the expected throughput by our burst allocator and the maximum throughput by an *ideal* one. In the ideal burst allocator, the number of IEs is equal to the number of resource assignments from the scheduler. Also, the frame resource is always allocated to urgent traffics (**P1**) first and then to non-real-time traffics (**P3**) with the highest transmission rate. It follows that two factors may degrade network throughput by our burst allocator: 1) extra IEs incurred by step 3 in Section 4.2 and 2) the data padding of low-rate non-real-time traffics at the boundary between the data in **P1** and **P3**. Specifically, 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 non-highest transmission rate. In the right-hand side of Fig. 3, it 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 above formulation, our objective is to find the throughput loss $\mathcal{L}$ by our burst allocator compared with the ideal one:

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

where $\tilde{\mathcal{O}}$ is the random variable representing the number of extra IEs caused by buckets and $\tilde{\mathcal{S}}$ is the random variable representing the throughput degradation (in bits) caused by the low-rate padding in the shaded area of the second possibility in 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.

5.1 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 totally six arrangement units, denoted by O_1, O_2, O_3, O_4, O_5, and O_6. Resources allocated to the three MSSs can be represented by two separators '|'. For example, we list 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, $\frac{Y}{\Delta_{bkt}} (= B)$ number of buckets, and $X \times B (= \alpha)$ arrangement units (*i.e.*, each bucket has X arrangement units). This 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 4. Whenever no separator appears at a bucket boundary, one extra IE is needed. There are totally $\frac{(\alpha+(n-1))!}{\alpha!(n-1)!}$ ways to place these separators. Let $\tilde{\mathcal{E}}$ be the random variable representing the number of bucket boundaries, where each of them 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} is the combinations to choose 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 Eq. (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, $i = 1..\alpha$. Explicitly, we have

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

However, when $\tilde{\mathcal{E}} = e$, $(B-1-e)$ of these x_i's must be 0. Also, 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

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

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

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

5.2 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 traffics. 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}. Since there is no throughput degradation by MSSs using the high transmission rate c_{high}, the overall expected throughput degradation is

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

Let $\tilde{U}_i$ be the random variable representing the data amount of M_i's urgent traffic, $i = 1..n$. 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 representing the amount of throughput degradation (in bits) due to the data padding of M_j's non-real-time traffic when using c_{low}. Since the throughput degradation caused by MSSs using c_{high} is zero, we have

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

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

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

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

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

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

By combining Eqs. (9), (10), (11), and (12), we can derive that

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

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

6 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 (physical) 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 traffics and feed them to the scheduling module for allocating resources and 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 module and the burst allocating module will interact with each other, especially for our scheme.

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

For the channel estimating module, we develop two scenarios to estimate the transmission rate of each MSS. The first scenario, called the *SUI (Stanford university interim) scenario*, is based on the SUI path loss model recommended by the 802.16 task group [24]. In particular, each MSS will roam inside the BS's signal coverage (which is 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 meters per second [25]. The transmission rate of each MSS M_i is determined by its received SNR (signal-to-noise ratio):

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

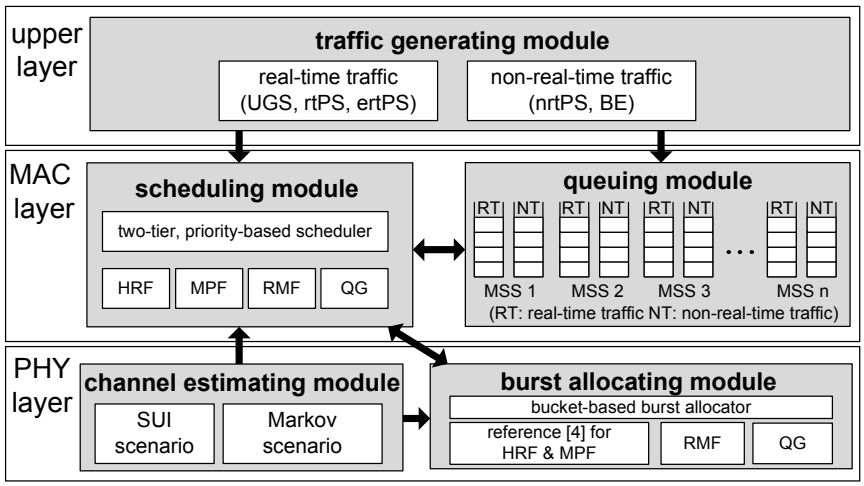


Fig. 6: The architecture of our C++ simulator.

TABLE 3: The 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

TABLE 4: The simulation parameters used in the SUI scenario

parameter	value
P_{BS}	1000 milliwatts
subchannel bandwidth (BW)	10 MHz
path loss model	SUI
Tx/Rx antenna gain	BS: 16 dBi/16 dBi; MSS: 8 dBi/8 dBi
antenna hight	BS: 30 meters; MSS: 2 meters
thermal noise	-100 dBm

where BW is the effective channel bandwidth (in Hz), 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 3. Specifically, the BS will choose the highest MCS whose minimum required SNR is smaller than $SNR(BS, M_i)$. Table 4 lists the parameters used in the SUI scenario.

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. Specifically, let $MCS[i]$ be the ith MCS, $i = 1..6$. 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 $\frac{1}{2}p_c$, 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.

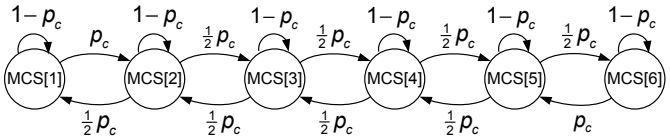


Fig. 7: A six-state Markov chain to model the channel condition.

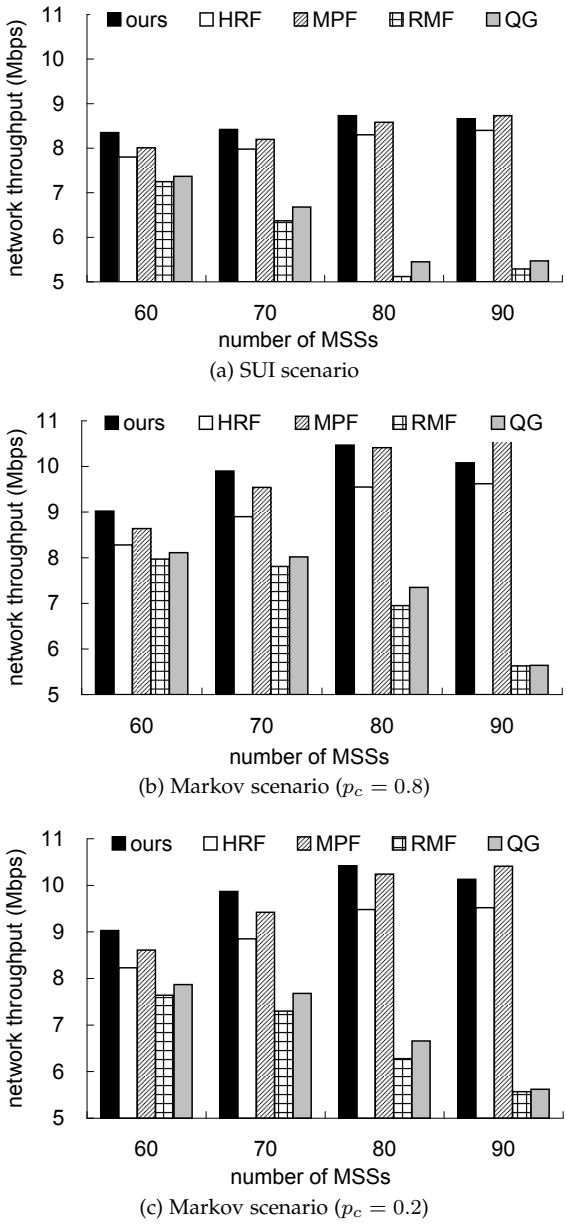
We compare our cross-layer framework against the *high rate first (HRF)* scheme [21], the *modified proportional fair (MPF)* scheme [10], the *rate maximization with fairness consideration (RMF)* scheme [20], and the *QoS guarantee (QG) scheme* [22]. HRF always first selects the MSS with the highest 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 those unsatisfied MSSs according to their minimum requirements, where MSSs are sorted by their transmission rates. If there remains resources, they are allocated to the MSSs with higher transmission rates. Similarly, QG first satisfies the minimum requirements of each MSS's traffics, which are divided into real-time and non-real-time ones. Then, the remaining resources are allocated to those MSSs with higher transmission rates. Since 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 6.6, we will discuss the effects of these two parameters on the system performance. The duration of each experiment is at least 2000 frames.

6.1 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 2000 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 steady when there are too many MSSs (*i.e.*, $n \geq 80$). The throughput under the SUI scenario is lower than that under the Markov scenario because some MSSs may move around the boundary of the BS's coverage,

leading a lower SNR and thus a lower MCS. Under the Markov scenario, a higher p_c means that each MSS may change its MCS more frequently and vice versa.

Generally speaking, both RMF and QG ignore the effect of IE overheads on network performance so that their throughput will be degraded. Although HRF serves those MSSs with higher transmission rates first, 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 $\frac{C_i}{C_i^{avg}}$ factor on potential throughput when scheduling traffics. The throughput of MPF is higher than that of RMF, QG, and HRF due to two reasons: First, MPF prefers those MSSs using 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 6.2). 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 traffics to reduce IE overheads while still guarantee long-term fairness; this will be verified in Sections VI-B and VI-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 reasons: First, our scheduler assigns a higher priority to those MSSs with higher C_i and $\frac{C_i}{C_i^{avg}}$ 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 traffics to avoid their packets being dropped. In this case, its throughput is slightly lower than that of MPF but our cross-layer framework can significantly reduce the real-time packet dropping ratio, as will be shown in Section 6.4.



(a) SUI scenario

(b) Markov scenario ($p_c = 0.8$)

(c) Markov scenario ($p_c = 0.2$)

Fig. 8: Comparison on network throughput.

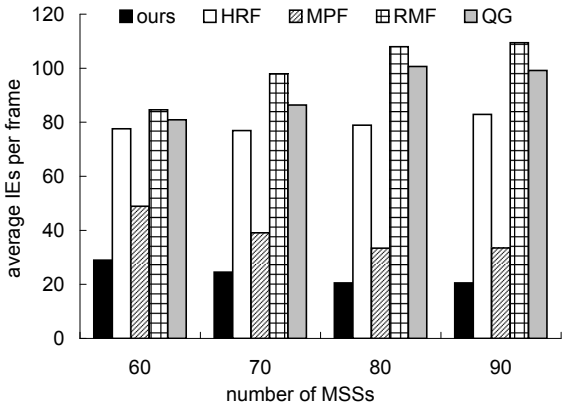
6.2 IE Overheads and Subframe Utilization

Fig. 9 shows the average number of IEs in each downlink subframe. As discussed earlier, HRF, RMF, and QG do not consider IE overheads so that they will generate a large number of IEs. The situation becomes worse when the number of MSSs grows, since 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). From Fig. 9, our cross-layer framework generates the smallest number of IEs per frame, because not only both the proposed scheduler and burst allocator do consider IE overheads, but also the framework can adjust the number of non-urgent real-time traffics to be served to avoid generating too many bursts.

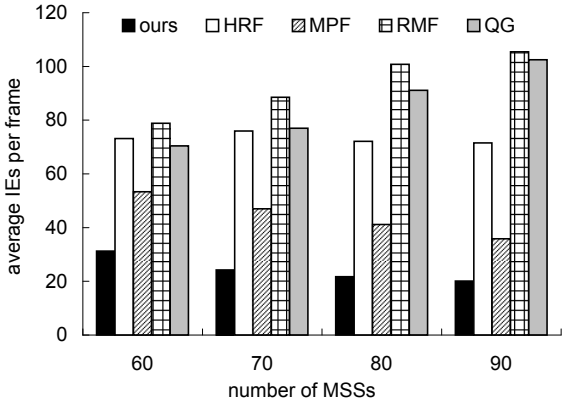
IE overheads have strong impact on the utilization of downlink subframes, as reflected in Fig. 10. Since HRF, RMF, and QG generate a large number of IEs, their subframe utilization will be lower than MPF and our cross-layer framework. It can be observed that the number of buckets B significantly affects the subframe utilization of our cross-layer framework. In particular, a too 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 too 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. From Fig. 10, we suggest setting $B = 5$ to get the best utilization and the analysis result in Section 6.7 will also validate this point.

6.3 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 of all schemes. Recall that the network becomes saturated when there are 80 MSSs. Thus, it is impossible to get a fairness index of one because the network resource is not enough to satisfy the requirements of all traffics. From Fig. 11, HRF incurs the lowest index because it always serves those MSSs using 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 and thus leading higher indices. Since RMF allocates the resources to MSSs sorted by their transmission rates, its index will be lower than QG.

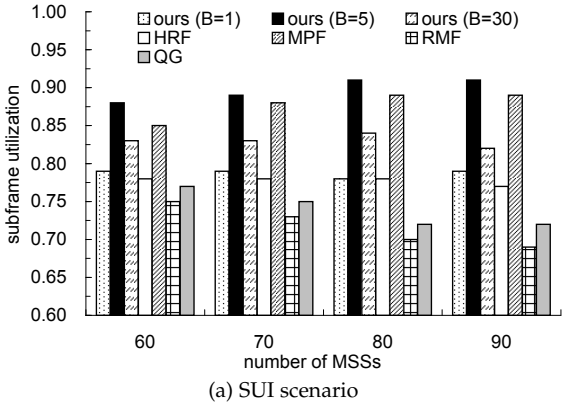


(a) SUI scenario

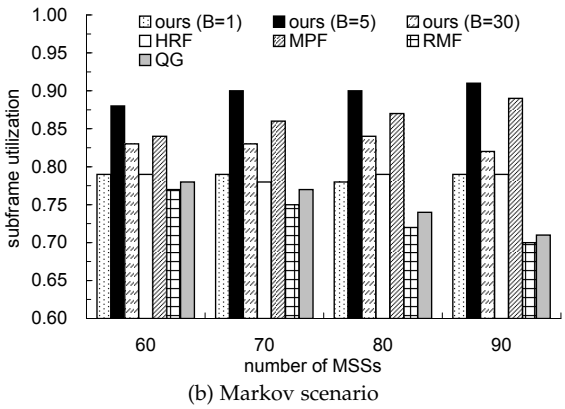


(b) Markov scenario

Fig. 9: Comparison on IE overheads.

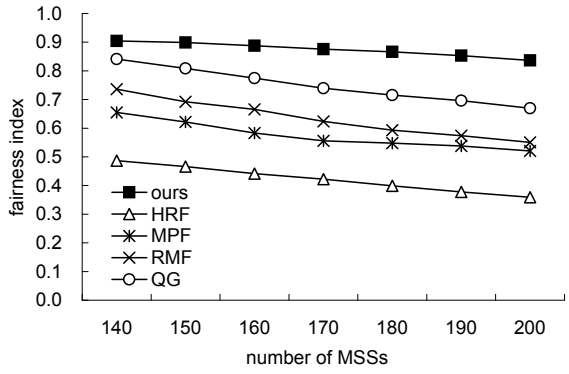


(a) SUI scenario

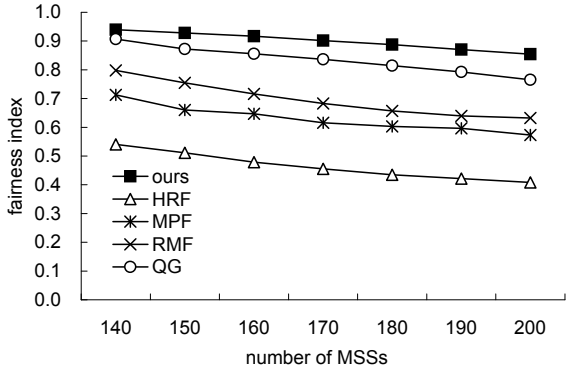


(b) Markov scenario

Fig. 10: Comparison on subframe utilization.



(a) SUI scenario



(b) Markov scenario

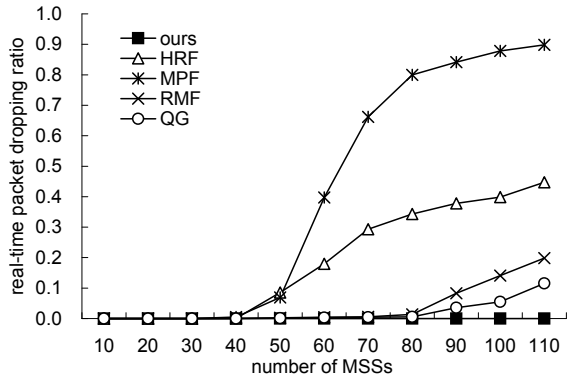
Fig. 11: Comparison on long-term fairness.

Our cross-layer framework has the highest fairness index (more than 0.85) due to two reasons: First, our priority-based scheduler only schedules γ ratio of non-urgent real-time traffics to avoid starving non-real-time traffics. Second, our cross-layer framework tries to reduce the IE overheads and acquire more frame space to allocate bursts for MSSs' traffics. 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.

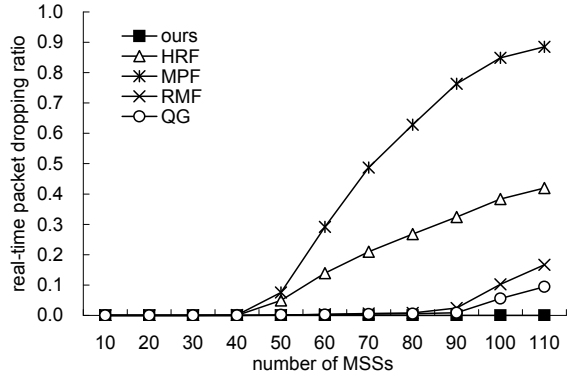
6.4 Packet Dropping Ratios of Real-Time Traffics

We then observe the packet dropping ratios of real-time traffics, where each MSS will generate $0 \sim 2R_i^{rt}$ amount of 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, so their ratios begin raising when $n \geq 50$. In this case, a large amount of non-real-time traffics will compete with real-time traffics for the limited resource. On the other hand, the ratios of RMF and QG begin raising when $n \geq 90$. Since both RMF and QG try to satisfy the minimum requirements of all traffics 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 three reasons: First, our priority-based scheduler assigns urgent real-time traffics with the highest priority. Also, it schedules a γ ratio of non-urgent real-time traffics to avoid generating too many urgent traffics in the following frames. Second, our bucket-based burst allocator arranges bursts based on the priorities from the scheduler, so

the bursts of those urgent real-time traffics can be allocated first to avoid packet dropping. Third, both our scheduler and burst allocator try to reduce IE overheads and thus more urgent real-time traffics can be served in each frame.



(a) SUI scenario



(b) Markov scenario

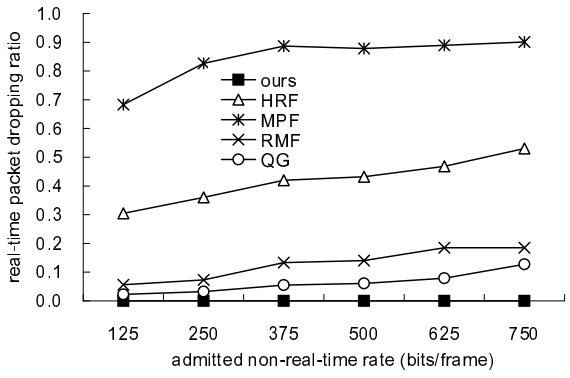
Fig. 12: Comparison on real-time packet dropping ratios under different number of MSSs.

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. Since MPF proportionally distributes resources among MSSs, it incurs the highest real-time packet dropping ratio. On the other hand, since some MSSs with real-time traffics may have higher transmission rates, the ratio of HRF is lower than that of MPF. As discussed earlier, both RMF and QG try to satisfy the minimum requirement of each traffic, their ratios can become lower. Note that since QG differentiates real-time traffics from non-real-time ones, its ratio is lower than that of RMF. Our cross-layer framework always has the zero ratio because not only the bursts of urgent real-time traffics are allocated first but also our framework can acquire more frame space to serve urgent real-time traffics by reducing IE overheads.

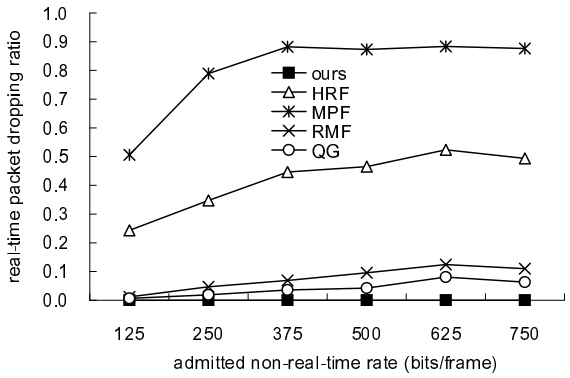
Since the trends under both SUI and Markov scenarios are similar, we only show the results under the Markov scenario in the following experiments.

6.5 Satisfaction Ratios of Non-Real-Time Traffics

Next, we measure the satisfaction ratios of non-real-time traffics (by Eq. (5)) under a saturated network. Fig. 14 shows the satisfaction ratios of non-real-time traffics 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



(a) SUI scenario



(b) Markov scenario

Fig. 13: Comparison on real-time packet dropping ratios under different admitted non-real-time rates.

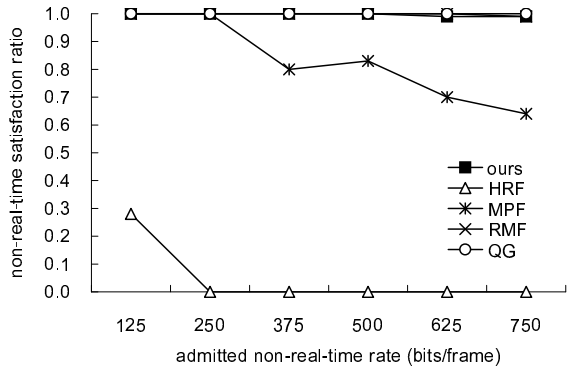


Fig. 14: Comparison on non-real-time satisfaction ratios of the bottom 10% MSSs under the Markov scenario.

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 traffics. 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 traffics will not be starved even though our scheme prefers real-time traffics.

6.6 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

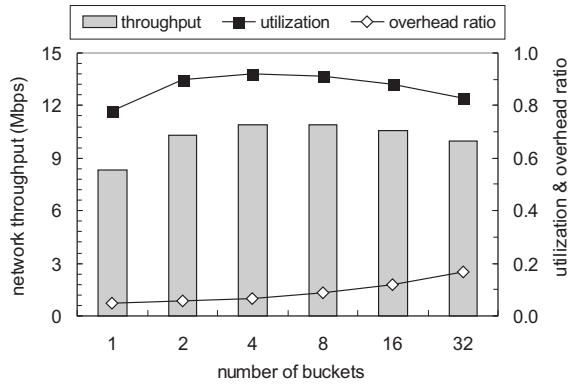


Fig. 15: Effect of the number of buckets (B) on network throughput, subframe utilization, and IE overheads under the Markov scenario.

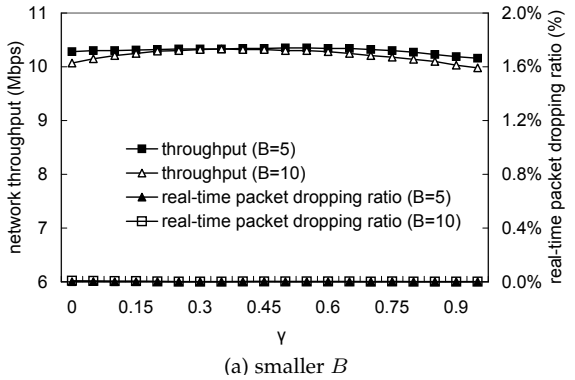
the ratio of the number of slots used for MAP information (*e.g.*, DL-MAP, UL-MAP, and IEs) to the total number of slots in a downlink subframe. Generally speaking, the utilization decreases when the overhead ratio increases, since they are complementary to each other. From Fig. 15, the utilization first increases and then decreases when B grows. The former increment is due to that some resource assignments do not fully utilize their allocated bursts. On the other hand, the later 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. From Fig. 15, we suggest setting $B = 4 \sim 8$ since this range of B value improves both throughput and utilization while reduces 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 traffics can be served. However, when γ increases, the throughput may decrease because the scheduler has to select more non-urgent real-time traffics to serve. In this case, some real-time traffics with lower transmission rates may be served, which degrades the throughput. As mentioned earlier, 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 that in the case of smaller B (*e.g.*, $B = 5$ and 10). From Fig. 16, we suggest setting $\gamma = 0.15 \sim 0.45$ since this range of γ value not only improves network throughput but also reduces real-time packet dropping ratios, under different values of B.

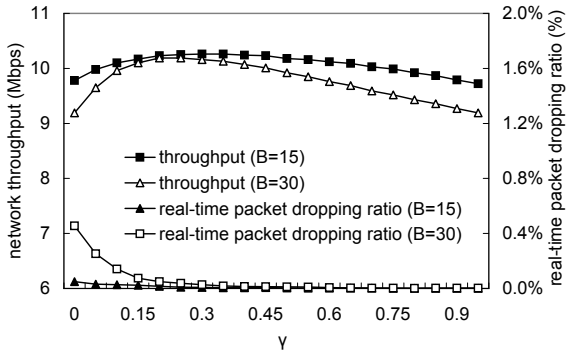
6.7 Verification of Throughput Analysis

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

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



(a) smaller B



(b) larger B

Fig. 16: Effect of γ on network throughput and real-time packet dropping ratios under the Markov scenario.

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

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

where T_D is the deadline of real-time data that will be dropped (in frames). Note that since 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 show 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 $B = [4, 8]$ while the maximum network throughput by simulation appears at the same range of $B = [4, 8]$. Thus, our analysis and simulation results are consistent. From 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.

7 Conclusions

In the 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 802.16 OFDMA network. Compared with existing solutions, our framework is more complete because it involves in co-designing of both the two-tier, priority-based scheduler and

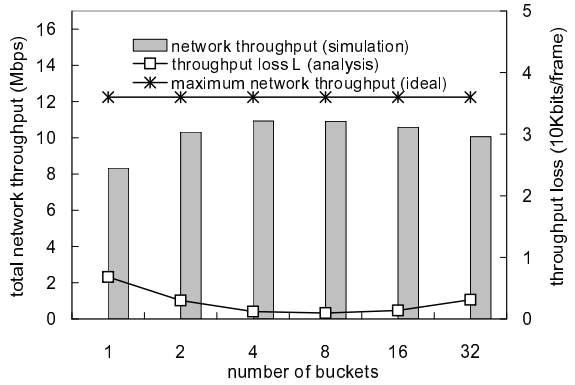


Fig. 17: Effect of the number of buckets (B) on the throughput loss $\mathcal{L}$ (by analysis) and the total network throughput (by simulation) under the Markov scenario.

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 rule, it guarantees real-time traffic delays, ensures satisfaction ratios of non-real-time traffics, and maintains long-term fairness. On the other hand, our burst allocator incurs low complexity and guarantees a bounded number, $\left(k + \frac{Y}{\Delta_{bkt}} - 1\right)$, of IEs to accommodate data bursts. In addition, it follows the priority rule from the scheduler to avoid packet dropping of urgent real-time traffics. We have also analyzed the impact of the number of buckets on the throughput loss. Through both analyses and simulations, we show how to adjust the system parameters to reduce IE overheads, improve subframe utilization, and enhance network throughput. Besides, these results also 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 to optimize the scheduler and burst allocator for some particular cases such as various traffic characteristics and MSS densities. In addition, we will consider extending our results for WiMAX relay networks.

REFERENCES

[1] IEEE Standard 802.16e-2005, "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," 2006.

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

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

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

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

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

[7] J. Shi and A. Hu, "Maximum utility-based resource allocation algorithm in the IEEE 802.16 OFDMA System," *Proc. IEEE Int'l Conf. Comm.*, pp. 311–316, 2008.

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

[9] N.A. Ali, M. Hayajneh, and H. Hassanein, "Cross layer scheduling algorithm for IEEE 802.16 broadband wireless networks," *Proc. IEEE Int'l Conf. Comm.*, pp. 3858–3862, 2008.

[10] J. Kim, E. Kim, and K.S. Kim, "A new efficient BS scheduler and scheduling algorithm in WiBro systems," *Proc. IEEE Int'l Conf. Advanced Comm. Technology*, pp. 1467–1470, 2006.

[11] T. Wang, H. Feng, and B. Hu, "Two-dimensional resource allocation for OFDMA system," *Proc. IEEE Int'l Conf. Comm. Workshops*, 2008.

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

[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," *Proc. IEEE Consumer Comm. and Networking Conf.*, pp. 890–896, 2008.

[14] A. Erta, C. Cicconetti, and L. Lenzini, "A downlink data region allocation algorithm for IEEE 802.16e OFDMA," *Proc. IEEE Int'l Conf. Information, Comm. & Signal Processing*, 2007.

[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 Comm. Magazine*, vol. 43, no. 12, pp. 136–146, 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 multi-frame structure for a periodic fixed bandwidth assignment scheme," *IEEE 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," *Proc. IEEE Vehicular Technology Conf.*, pp. 284–288, 2007.

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

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

[21] X. Zhu, J. Huo, S. Zhao, Z. Zeng, and W. Ding, "An adaptive resource allocation scheme in OFDMA based multiservice WiMAX Systems," *Proc. IEEE Int'l Conf. Advanced Comm. Technology*, pp. 593–597, 2008.

[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," *Proc. IEEE Int'l Conf. Comm.*, pp. 3463–3468, 2008.

[23] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *J. Computer Networks and ISDN*, vol. 17, no. 1, pp. 1–14, 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 Computing*, vol. 2, no. 3, pp. 257–269, 2003.

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