Exploiting Spectral Reuse in Routing, Resource Allocation, and Scheduling for IEEE 802.16 Mesh Networks

Lien-Wu Chen, Yu-Chee Tseng, *Senior Member, IEEE*, You-Chiun Wang, *Member, IEEE*, Da-Wei Wang, *Member, IEEE*, and Jan-Jan Wu, *Member, IEEE*

***Abstract*—The IEEE 802.16 standard for wireless metropolitan area networks (WMANs) is defined to meet the need for wide-range broadband wireless access at low cost. The objective of this paper is to study how to exploit spectral reuse in resource allocation in an IEEE 802.16 mesh network, which includes routing tree construction (RTC), bandwidth allocation, time-slot assignment, and bandwidth guarantee of real-time flows. The proposed spectral reuse framework covers bandwidth allocation at the application layer, RTC and resource sharing at the medium access control (MAC) layer, and channel reuse at the physical layer. To the best of our knowledge, this is the first paper that formally quantifies spectral reuse in IEEE 802.16 mesh networks and exploits spectral efficiency under an integrated framework. Simulation results show that the proposed schemes significantly improve the throughput of IEEE 802.16 mesh networks.**

***Index Terms*—IEEE 802.16, mesh network, resource allocation, routing tree, WiMax, wireless network.**

I. Introduction

TO ACHIEVE the requirement of wide-range wireless broadband access at a low cost, the IEEE 802.16 standard [1] has recently been proposed. The goal of this standard is to solve the last-mile problem in a metropolitan area network in a more flexible and economical way as opposed to traditional cabled access networks, such as fiber optics, digital subscriber line (DSL), or T1 links [2], [3]. The IEEE 802.16 standard is based on a common medium access control (MAC) protocol that is compliant with different physical layer specifications. The physical layer can employ the orthogonal frequency-division multiplexing (OFDM) scheme below 11 GHz or the single-carrier scheme between 10 and 66 GHz.

The IEEE 802.16 MAC protocol supports the *point-to-multipoint (PMP)* mode and the *mesh* mode. In PMP mode, stations are organized as a cellular network, where subscriber stations (SSs) are directly connected to base stations (BSs). Such networks require each SS to be within the communication range of its associated BS, thus greatly limiting the coverage range of the network. On the other hand, in mesh mode, stations are organized in an ad hoc fashion. Each SS can act either as an end point or as a router to relay traffics for its neighbors. Thus, there is no need to have a direct link from each SS to its associated BS. This leads to two advantages: SSs may transmit at higher rates to their parent SSs or BS, and a BS can serve a wider coverage at a lower deployment cost [4].

In an IEEE 802.16 mesh network, transmissions can undergo a multihop manner. The standard specifies a centralized scheduling mechanism for the BS to manage the network. Stations will form a *routing tree* rooted at the BS for the communication purpose. SSs in the network will send request messages containing their traffic demands and link qualities to the BS to ask for resources. The BS then uses the topology information along with SSs' requests to determine the routing tree and to allocate resources. Resources in an IEEE 802.16 network are usually represented by time slots within a frame. Our goal is to solve the *resource allocation problem* given the uplink/downlink bandwidth demands of each SS and their link qualities. There are four issues to be considered.

1) Tree reconstruction: How do we determine the routing tree based on SSs' current bandwidth demands and link qualities?
2) Bandwidth allocation: How do we determine the number of time slots of each SS according to its uplink and downlink bandwidth demands?
3) Time-slot assignment: How do we assign time slots to each SS in a frame?
4) Bandwidth guarantee: How do we schedule transmission on time slots for each SS so that a fixed amount of bandwidth is guaranteed for each real-time flow?

In this paper, we investigate the resource-allocation problem by exploring the concept of *spectral reuse*. Although it is well known that a time slot used by a station can be "reused"

Manuscript received August 7, 2007; revised January 1, 2008, February 26, 2008, and March 28, 2008. First published April 18, 2008; current version published January 16, 2009. The work of Y.-C. Tseng was supported in part by Taiwan MoE ATU plan, by the NSC under Grant 93-2752-E-007-001-PAE, Grant 96-2623-7-009-002-ET, Grant 95-2221-E-009-058-MY3, Grant 95-2221-E-009-060-MY3, Grant 95-2219-E-009-007, Grant 95-2218-E-009-209, and Grant 94-2219-E-007-009, by Realtek Semiconductor Corporation, by MOEA under Grant 94-EC-17-A-04-S1-044, by ITRI, Taiwan, by Microsoft Corporation, and by Intel Corporation. The review of this paper was coordinated by Dr. H. Jiang.

L.-W. Chen is with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan, and also with Academia Sinica, Taipei 115, Taiwan (e-mail: lwchen@cs.nctu.edu.tw).

Y.-C. Tseng is with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan, and also with Chung Yuan Christian University, Chung Li 32023, Taiwan (e-mail: yctseng@cs.nctu.edu.tw).

Y.-C. Wang is with the Department of Computer Science, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: wangyc@cs.nctu.edu.tw).

D.-W. Wang and J.-J. Wu are with the Institute of Information Science, Academia Sinica, Taipei 11529, Taiwan (e-mail: wdw@iis.sinica.edu.tw; wuj@iis.sinica.edu.tw).

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.2008.923685

TABLE I
COMPARISON OF PRIOR WORKS [15]–[17] AND OUR SPECTRAL REUSE FRAMEWORK

features	reuse modeling[1]	load awareness	tree reconstruction	time-slot allocation	bandwidth guarantee[3]
reference [15]			partial[2]	√	
reference [16]			partial[2]		
reference [17]			√	√	
our framework	√	√	√	√	√

[1] Mathematical modeling is provided to evaluate the degree of spectral reuse.
[2] Initial tree construction is provided, but without tree reconstruction.
[3] The guarantee is for real-time flows.

by another station if the latter is sufficiently separated from the former, the IEEE 802.16 standard does not explore this. We propose a spectral reuse framework to efficiently allocate resources in an IEEE 802.16 mesh network with global fairness in mind, that is, the bandwidths allocated to SSs will be proportionate to their requests in an end-to-end (SS-to-BS) sense. Our framework includes routing tree construction (RTC) and a centralized scheduling algorithm. The former allows a BS to form an efficient routing tree according to SSs' bandwidth demands and interferences. The latter helps a BS determine bandwidth allocation and time-slot assignment. In particular, when time slots are tight, we show how to adjust scheduling to prioritize real-time from nonreal-time traffic to guarantee some bandwidth for real-time traffic. Note that the tree topology is consistent with the current IEEE 802.16 standard. In addition, our framework does not require any change to the message structures and the signaling mechanism defined in the standard.

In the literature, early works on the IEEE 802.16 standard have primarily focused on the PMP mode [5]–[7]. For the mesh mode, former efforts have been devoted to topology design [8], packet scheduling [9], [10], and QoS support [11], [12]. Reference [13] shows how to manage radio resources in a WiMax single-carrier network in a distributed manner. Reference [14] discusses how to improve channel efficiency and provide fair access to SSs. The BS allocates time slots to SSs in a per-hop basis in such a way that one-hop nodes will have precedence over two-hop nodes ("hop" in the sense of nodes' distances to the BS). Similarly, i-hop nodes will have precedence over $(i+1)$-hop nodes. However, this may lead to starvation of farther away SSs as the network becomes congested, particularly when SSs with smaller hop counts request larger bandwidths. On the contrary, our scheduling algorithm allocates time slots to SSs that are proportionate to their requests and thus avoids such starvation.

Several studies [15]–[17] have addressed the issue of spectral reuse to solve the resource allocation problem. Reference [15] proposes RTC and a scheduling algorithm by considering interference among neighboring SSs. It attempts to find a route to reduce the interference among SSs and then to maximize the number of concurrent transmissions. How to attach a new SS to a routing tree incurring the least interference is discussed in [16]. In [17], the authors indicate that the network performance highly depends on the order that SSs join the routing tree and then propose a routing tree reconstruction and concurrent transmission scheme to achieve spectral reuse. As can be seen, the prior works only discuss partial aspects of the resource-allocation problem.

Table I compares the functions provided by other schemes and ours. Our framework offers the most complete solution to the resource-allocation problem. The contributions of our framework are fourfold. First, it formally quantifies spectral reuse in a mesh network and is thus capable of achieving higher spectral efficiency. Second, it takes dynamic traffic demands of SSs into account and includes not only a tree optimization algorithm but bandwidth allocation and time-slot assignment as well. Third, we propose a way to prioritize real-time from nonreal-time traffic so that a fixed amount of bandwidth is maintained for each real-time flow when resources are stringent. Finally, the proposed framework covers bandwidth allocation at the application layer, RTC and resource sharing at the MAC layer, and channel reuse at the physical layer. Extensive performance studies are conducted, and the simulation results show that our framework can achieve better spectral reuse and higher network throughput compared with existing results.

The rest of this paper is organized as follows. Section II briefly reviews the operations of an IEEE 802.16 mesh network and formally defines the resource-allocation problem. Section III proposes our spectral reuse framework. Section IV discusses how to guarantee the bandwidth of real-time traffic by our framework. Section V gives the simulation results. Section VI concludes this paper.

II. PRELIMINARY

A. Resource Allocation in an IEEE 802.16 Mesh Network

An IEEE 802.16 mesh network is composed of a BS and several SSs. These stations form a routing tree rooted at the BS, and transmissions between stations may undergo a multihop manner. The IEEE 802.16 MAC protocol supports both centralized and distributed scheduling methods. In this paper, we focus on centralized scheduling to fully exploit spectral reuse.

In centralized scheduling, the standard supports two control messages, i.e., *Mesh Centralized Scheduling Configuration (MSH-CSCF) and Mesh Centralized Scheduling (MSH-CSCH)*, to help the BS establish its routing tree and specify the transmission schedules of SSs in the network. To achieve this, the BS first broadcasts an MSH-CSCF message containing the routing tree information to the network. An SS receiving such a message can know its parent and children in the tree and then rebroadcasts the MSH-CSCF message according to its index specified in the message. This procedure is repeated until all SSs have received the MSH-CSCF message.

After constructing the routing tree by the MSH-CSCF message, SSs can transmit MSH-CSCH:Request messages to

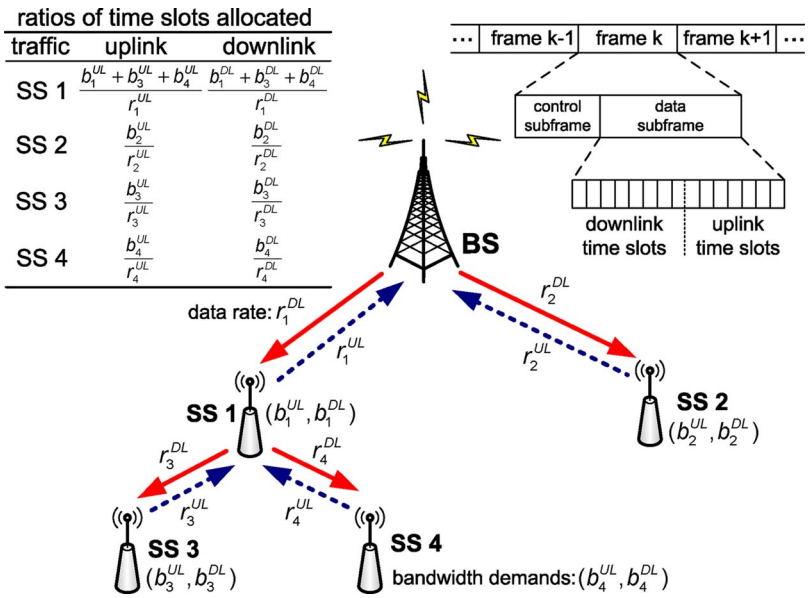


Fig. 1. Bandwidth allocation example in the IEEE 802.16 standard.

request time slots. The transmission order is from leaves to the root. An SS will combine the requests from its children into its own MSH-CSCH:Request message and then transmits the message to its parent. This way, the BS can gather bandwidth requests from all SSs and then broadcasts an MSH-CSCH:Grant message containing the slot allocations to all SSs. Note that the BS can also update the routing tree by containing the tree update information in the MSH-CSCH:Grant message. In this case, SSs have to update their positions in the new tree according to the message. Otherwise, the routing tree remains the same as specified in the previous MSH-CSCF message. Note that according to the 802.16 standard, the period during which the MSH-CSCH schedule is valid is limited by the time that the BS takes to aggregate traffic requirements and distribute the next schedule. Therefore, the scheduling interval is several frames, depending on the size of the mesh network. Therefore, it is reasonable to assume that the link data rates and bandwidth demands of SSs are constants over a short period of time.

To allocate bandwidths for SSs, the IEEE 802.16 standard gives an example, as illustrated in Fig. 1. Each SS i first sends its uplink bandwidth demand b_i^{UL} and downlink bandwidth demand b_i^{DL} to the BS. Let the uplink and downlink data rates of SS i be r_i^{UL} and r_i^{DL}, respectively. The ratios of uplink slots allocated to SS 1, SS 2, SS 3, and SS 4 will be $(b_1^{\mathrm{UL}} + b_3^{\mathrm{UL}} + b_4^{\mathrm{UL}})/r_1^{\mathrm{UL}} : \ b_2^{\mathrm{UL}}/r_2^{\mathrm{UL}} : \ b_3^{\mathrm{UL}}/r_3^{\mathrm{UL}} : b_4^{\mathrm{UL}}/r_4^{\mathrm{UL}} (= \gamma_1 : \ \gamma_2 : \ \gamma_3 : \ \gamma_4)$. Note that here the calculation also includes the relay traffic. If $N_{\mathrm{total}}^{\mathrm{UL}}$ is the total number of uplink slots per frame, the numbers of slots allocated to them are $\gamma_1 \cdot N_{\mathrm{total}}^{\mathrm{UL}} / \sum_{i=1}^{4} \gamma_i$, $\gamma_2 \cdot N_{\mathrm{total}}^{\mathrm{UL}} / \sum_{i=1}^{4} \gamma_i$, $\gamma_3 \cdot N_{\mathrm{total}}^{\mathrm{UL}} / \sum_{i=1}^{4} \gamma_i$, and $\gamma_4 \cdot N_{\mathrm{total}}^{\mathrm{UL}} / \sum_{i=1}^{4} \gamma_i$, respectively. The bandwidth allocation for downlink traffic follows the same way.

However, the above bandwidth allocation is very inefficient because a slot is always allocated to only one SS. In fact, SS 2 and SS 3 can concurrently transmit without interfering with each other. We can quantify the waste of slots as follows. Given a routing tree $\mathcal{T}$, the *aggregated uplink bandwidth demand* d_i^{UL} for each SS i is defined as

$$d_i^{\mathrm{UL}} = b_i^{\mathrm{UL}} + \sum_{j \in \mathrm{child}(i)} d_j^{\mathrm{UL}} \tag{1}$$

where $\mathrm{child}(i)$ is the set of SS i's children in $\mathcal{T}$. Then, the *demand of uplink transmission time* for SS i is

$$T_i^{\mathrm{UL}} = \frac{d_i^{\mathrm{UL}}}{r_i^{\mathrm{UL}}}. \tag{2}$$

Let us denote the sum of the uplink transmission time of all SSs by

$$C_{\mathrm{total}}^{\mathrm{UL}} = \sum_{i \in \mathcal{T} - \mathrm{BS}} T_i^{\mathrm{UL}}.$$

Therefore, only a ratio of $T_i^{\mathrm{UL}}/C_{\mathrm{total}}^{\mathrm{UL}}$ of the uplink slots is allocated to SS i. However, let the sum of the transmission times of SS i and its interference neighbors be

$$C_i^{\mathrm{UL}} = \sum_{j \in E_i} T_j^{\mathrm{UL}} \tag{3}$$

where $E_i = \{i\} \cup \mathcal{I}(i)$, and $\mathcal{I}(i)$ is the set of interference neighbors of SS i. From SS i's perspective, it only sees a ratio of $C_i^{\mathrm{UL}}/C_{\mathrm{total}}^{\mathrm{UL}}$ of the uplink slots to be busy. In other words, the remaining $1 - (C_i^{\mathrm{UL}}/C_{\mathrm{total}}^{\mathrm{UL}})$ portion of time is simply idle, as seen by SS i. The downlink direction will suffer from a similar waste.

B. Problem Definition

The problem with the preceding waste is due to the lack of spectral reuse. Our goal is to solve the resource-allocation problem in an IEEE 802.16 mesh network with spectral reuse. Given the uplink and downlink bandwidth demands b_i^{UL} and

TABLE II
SUMMARY OF NOTATIONS

notation	definition
N	number of time slots within a data subframe
$N_{\text{total}}^{\text{UL}}/N_{\text{total}}^{\text{DL}}$	number of uplink/downlink slots within a frame
$N_i^{\text{UL}}/N_i^{\text{DL}}$	number of uplink/downlink slots allocated to SS i
$b_i^{\text{UL}}/b_i^{\text{DL}}$	individual bandwidth demand of uplink/downlink traffics generated by SS i
$d_i^{\text{UL}}/d_i^{\text{DL}}$	aggregated bandwidth demands of uplink/downlink traffics delivered by SS i
$r_i^{\text{UL}}/r_i^{\text{DL}}$	uplink/downlink data rate of SS i
$T_i^{\text{UL}}/T_i^{\text{DL}}$	demand of uplink/downlink transmission time of SS i
E_i	set of SSs that contains SS i and its interference neighborhood $\mathcal{I}(i)$
$C_i^{\text{UL}}/C_i^{\text{DL}}$	aggregated $T_j^{\text{UL}}/T_j^{\text{DL}}$ of all SS j in E_i
$C_{\text{total}}^{\text{UL}}/C_{\text{total}}^{\text{DL}}$	aggregated $T_j^{\text{UL}}/T_j^{\text{DL}}$ of all SS j in the network
$C_{\max}^{\text{UL}}/C_{\max}^{\text{DL}}$	maximal $C_i^{\text{UL}}/C_i^{\text{DL}}$ among all SS i in the network

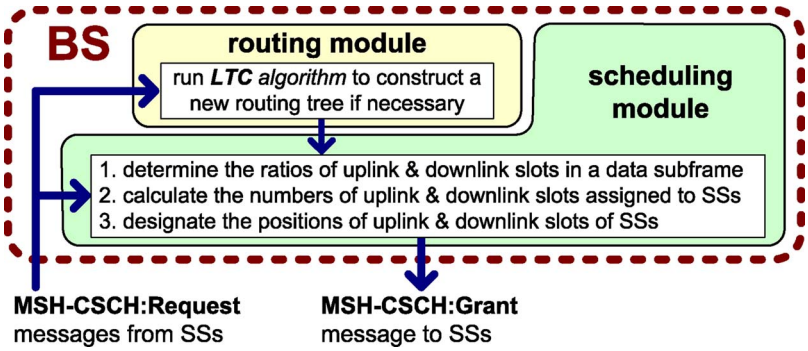


Fig. 2. System architecture of our spectral reuse framework.

b_i^{DL} and data rates r_i^{UL} and r_i^{DL}, respectively, of each SS i, we will consider the following four issues.

1) Tree reconstruction: How do we organize the routing tree according to SSs' bandwidth demands and data rates so that the traffic loads among tree nodes can be balanced and the network throughput can be maximized?
2) Bandwidth allocation: How do we allocate time slots to SSs according to their bandwidth demands and data rates so that SSs can fully utilize the channel?
3) Time-slot assignment: How do we assign slots of a frame for SSs with global fairness in mind so that the transmissions between SSs will not conflict with each other?
4) Bandwidth guarantee: How do we schedule real-time and nonreal-time traffic when resources are stringent so that the bandwidth requirements of real-time flows can be maintained?

III. SPECTRAL REUSE FRAMEWORK

In this section, we propose our spectral reuse framework to solve the first three issues in the resource-allocation problem. In Section IV, we will discuss how to extend our framework to provide bandwidth guarantee for real-time flows. Table II summarizes the notations used in this paper. Fig. 2 shows the system architecture of our framework. First, the BS collects the MSH-CSCH:Request messages and passes the bandwidth demands and data rates of SSs to the scheduling and routing modules. The scheduling module is a fast process that determines the number of time slots and their positions allocated to each SS in each frame. The routing module is a slow process, which continuously monitors the quality of the routing tree and reconstructs the tree when the quality of the tree degrades. That is, when it is found that the tree cannot efficiently deliver the traffics of SSs, a new routing tree will be computed by the routing module. The BS then broadcasts an MSH-CSCH:Grant message containing the new routing tree and time-slot allocation of each SS to the network.

In the following sections, we first present the basic concept of our spectral reuse framework, followed by the designs of the scheduling and routing modules.

A. Basic Concept

Earlier, we have indicated that, in the uplink case, the scheduling scheme in IEEE 802.16 only assigns $p_i = T_i^{\text{UL}}/C_{\text{total}}^{\text{UL}}$ portion of uplink slots to each SS i. From each SS i's view, the remaining $1 - (C_i^{\text{UL}}/C_{\text{total}}^{\text{UL}})$ portion of uplink slots are idle. Ideally, SS i may expect the idle portion to be fairly distributed to all SSs in E_i proportionally. This implies that SS i can share an additional $q_i = (1 - C_i^{\text{UL}}/C_{\text{total}}^{\text{UL}}) \times T_i^{\text{UL}}/C_i^{\text{UL}}$ portion of uplink transmission time. Thus, the total portion of uplink transmission time assigned to SS i is

$$\frac{T_i^{\text{UL}}}{C_{\text{total}}^{\text{UL}}} + \left(1 - \frac{C_i^{\text{UL}}}{C_{\text{total}}^{\text{UL}}}\right) \times \frac{T_i^{\text{UL}}}{C_i^{\text{UL}}} = \frac{T_i^{\text{UL}}}{C_i^{\text{UL}}}. \tag{4}$$

Similarly, the total portion of downlink transmission time assigned to SS i can be upgraded, ideally, to $T_i^{\text{DL}}/C_i^{\text{DL}}$.

Unfortunately, (4) does not consider the congestion issue in the global network. In a noncongested network, the uplink bandwidth of an SS should be able to deliver all traffic from itself plus those from its children. Otherwise, congestion on

that SS's uplink will occur. Therefore, given a noncongested network, if an SS i's uplink bandwidth is increased by a ratio of α, a sufficient condition to avoid the network becoming congested is to enforce the parent of SS i to increase its uplink bandwidth by at least a ratio of α. Now, let α_i be the ideal ratio of increase by SS i in the uplink direction

$$\alpha_i = \frac{q_i}{p_i} = \frac{\left(1 - \frac{C_i^{\mathrm{UL}}}{C_{\mathrm{total}}^{\mathrm{UL}}}\right) \times \frac{T_i^{\mathrm{UL}}}{C_i^{\mathrm{UL}}}}{\frac{T_i^{\mathrm{UL}}}{C_{\mathrm{total}}^{\mathrm{UL}}}} = \frac{C_{\mathrm{total}}^{\mathrm{UL}}}{C_i^{\mathrm{UL}}} - 1.$$

The minimum ratio of increase among all SSs is

$$\alpha_{\min} = \min_{\forall i}\{\alpha_i\} = \frac{C_{\mathrm{total}}^{\mathrm{UL}}}{C_{\max}^{\mathrm{UL}}} - 1 \geq 0$$

where $C_{\max}^{\mathrm{UL}} = \max_{\forall i}\{C_i^{\mathrm{UL}}\}$. Therefore, using $\alpha_{\min}$ as the global ratio of increase, the portion of uplink transmission time for each SS i such that the network will not be congested is

$$(1 + \alpha_{\min}) \times \frac{T_i^{\mathrm{UL}}}{C_{\mathrm{total}}^{\mathrm{UL}}} = \frac{T_i^{\mathrm{UL}}}{C_{\max}^{\mathrm{UL}}}.$$

Similarly, the portion of downlink transmission time for each SS i such that the network will not be congested is $T_i^{\mathrm{DL}}/C_{\max}^{\mathrm{DL}}$, where $C_{\max}^{\mathrm{DL}} = \max_{\forall i}\{C_i^{\mathrm{DL}}\}$.

Note that the above calculation includes the demands of individual SSs as well as relay traffic. Therefore, our slot allocation is in an end-to-end sense. Next, we discuss how to adopt this concept to the scheduling module to increase the channel efficiency. The routing module will reconstruct the routing tree to further improve the performance of the scheduling module. For readability, we first discuss how the scheduling module works and then present how the routing module works.

B. Scheduling Module

Given a routing tree $\mathcal{T}$, the scheduling module should properly allocate time slots to SSs in each frame so that the transmissions of nearby SSs will not cause collision, and global fairness among SSs can be maintained. Assuming N to be the total number of slots in a data subframe, the scheduling module involves the following steps.

1) We first choose the ratio of the number of uplink slots to the number of downlink slots to be $C_{\max}^{\mathrm{UL}} : C_{\max}^{\mathrm{DL}}$. Thus, the numbers of uplink and downlink slots in a data subframe observed by the BS are $N_{\mathrm{total}}^{\mathrm{UL}} = \lfloor C_{\max}^{\mathrm{UL}}/C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}} \times N \rfloor$ and $N_{\mathrm{total}}^{\mathrm{DL}} = \lfloor C_{\max}^{\mathrm{DL}}/C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}} \times N \rfloor$, respectively.[1]
2) Based on $N_{\mathrm{total}}^{\mathrm{UL}}$ and $N_{\mathrm{total}}^{\mathrm{DL}}$, we then allocate $N_i^{\mathrm{UL}} = T_i^{\mathrm{UL}}/C_{\max}^{\mathrm{UL}} \times N_{\mathrm{total}}^{\mathrm{UL}}$ and $N_i^{\mathrm{DL}} = T_i^{\mathrm{DL}}/C_{\max}^{\mathrm{DL}} \times N_{\mathrm{total}}^{\mathrm{DL}}$ slots to each SS i for its uplink and downlink traffics, respectively. Note that since spectral reuse is considered, it is possible that $\sum_{\forall i} N_i^{\mathrm{UL}} > N_{\mathrm{total}}^{\mathrm{UL}}$ and $\sum_{\forall i} N_i^{\mathrm{DL}} > N_{\mathrm{total}}^{\mathrm{DL}}$.
3) Next, we need to allocate N_i^{UL} collision-free uplink slots in each data subframe to SS i. These slots are divided into two parts. Part-1 contains $T_i^{\mathrm{UL}}/C_{\mathrm{total}}^{\mathrm{UL}} \times N_{\mathrm{total}}^{\mathrm{UL}}$ slots. Part-2 contains $(T_i^{\mathrm{UL}}/C_{\max}^{\mathrm{UL}} - T_i^{\mathrm{UL}}/C_{\mathrm{total}}^{\mathrm{UL}}) \times N_{\mathrm{total}}^{\mathrm{UL}}$ slots. Part-1 slots are more suitable for real-time traffics because a packet issued by any SS in $\mathcal{T}$ can be delivered to the BS with a latency of no more than one frame time (the reason will be explained in Theorem 1). Now, we describe how these slots are determined.
   a) Part-1 slots: These slots are assigned in a bottom-up manner along the tree $\mathcal{T}$. Specifically, we traverse SSs in $\mathcal{T}$ according to the transmission order of MSH-CSCH:Request messages. In IEEE 802.16, such an order is reverse in hop count to the BS (that is, largest hop count first) and is retained as the nodes' IDs in the routing tree for SSs with the same hop count. Thus, the order of a child SS is always before that of its parent. Following this transmission order, for each SS i being visited, we select the first $T_i^{\mathrm{UL}}/C_{Total}^{\mathrm{UL}} \times N_{Total}^{\mathrm{UL}}$ unoccupied slots as its part-1 slots and then mark these slots as *occupied*. This operation is repeated until all SSs are visited.
   b) Part-2 slots: We also assign these slots following the transmission order of MSH-CSCH:Request messages. For every SS i being visited, each of its part-2 slots is selected from the first unoccupied slot by any SS in E_i. Then, that slot is marked as occupied. The above operation is repeated until all SSs are visited.

   Algorithm 1 gives the pseudocode of the foregoing time-slot assignment scheme.

Algorithm 1: *Time-slot assignment for uplink traffics*

Input: numbers of uplink slots for SSs, $\{N_1^{\mathrm{UL}}, \cdots, N_n^{\mathrm{UL}}\}$
Output: result of slot assignment, *transmit*$[n][N_{\mathrm{total}}^{\mathrm{UL}}]$

```
// assign part-1 slots
let SS 1, 2, ..., n be the transmission order of
MSH-CSCH:Request messages in T;
free ← 1;
for i = 1 to n do
    allocated ← free + (T_i^UL / C_total^UL) × N_total^UL;
    for j = free to allocated do slot[j] ← i;
    free ← allocated;

// assign part-2 slots
for i = 1 to n do
    for j = 1 to N_total^UL do
        transmit[i][j] ← NULL;
for i = 1 to n do // mark occupied slots of SSs
    for j = 1 to N_total^UL do
        if slot[j] ∈ E_i then transmit[i][j] ← slot[j];
for i = 1 to n do
    allocated = (T_i^UL / C_max^UL − T_i^UL / C_total^UL) × N_total^UL;
    for j = 1 to N_total^UL do
        if allocated > 0 and transmit[i][j] = NULL then
            transmit[i][j] ← i;
            allocated ← allocated − 1;
            for k = 1 to n do
                if k ∈ E_i then transmit[k][j] ← i;
```

[1]Recall that $C_{\max}^{\mathrm{UL}}$ and $C_{\max}^{\mathrm{DL}}$ represent the maximum uplink and downlink demands, respectively, seen by individual nodes. They are bottlenecks of uplink and downlink transmissions. Therefore, we use the ratio of $C_{\max}^{\mathrm{UL}}$ and $C_{\max}^{\mathrm{DL}}$ to reflect the demands of uplink and downlink slots and use this ratio to distribute slots. Later on, we will construct the routing tree by minimizing the sum of $C_{\max}^{\mathrm{UL}}$ and $C_{\max}^{\mathrm{DL}}$ to improve spectral reuse. In addition, note that the number of slots should be bounded to integers. However, in the following, we will avoid using floor and ceiling functions for ease of presentation.

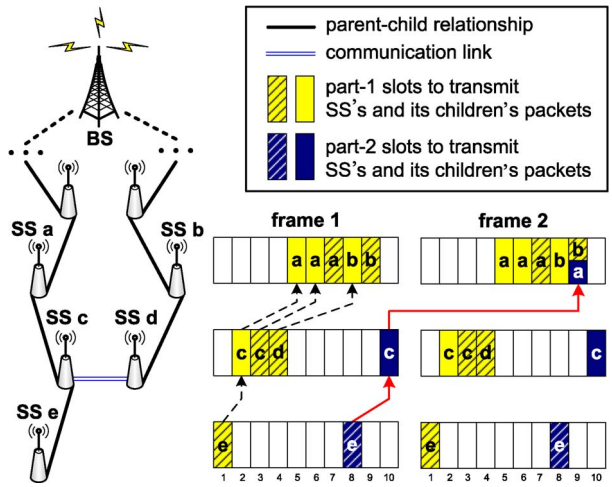


Fig. 3. Example of time-slot assignment for uplink traffics.

4) We then designate N_i^{DL} collision-free downlink slots to each SS i. These slots are also divided into two parts, where part-1 contains $T_i^{\mathrm{DL}}/C_{\mathrm{total}}^{\mathrm{DL}} \times N_{\mathrm{total}}^{\mathrm{DL}}$ slots and part-2 contains $(T_i^{\mathrm{DL}}/C_{\mathrm{max}}^{\mathrm{DL}} - T_i^{\mathrm{DL}}/C_{\mathrm{total}}^{\mathrm{DL}}) \times N_{\mathrm{total}}^{\mathrm{DL}}$ slots. For each part, we assign their slots in a top-down manner along the tree $\mathcal{T}$. Specifically, we traverse SSs in $\mathcal{T}$ by the transmission order of MSH-CSCH:Request messages and then assign slots to these SSs following the reverse order. For each SS being visited, we assign downlink slots to them according to the rules specified in step 3.

Consider an illustrative example in Fig. 3, where we need to assign uplink slots for five SSs in the network. Let the demand of each of the SSs a, b, c, and d be one slot, and let the demand of SS e be two slots. We assume that the interference neighborhood of an SS contains all its neighbors within the two-hop range. First, part-1 slots can easily be assigned in a sequential manner $(e \rightarrow c \rightarrow d \rightarrow a \rightarrow b)$. To assign part-2 slots, observe that the interference neighborhood $\mathcal{I}(a)$ of a includes c, d, and e. For e, we assign slot 8 as its part-2 slot since it is the first unoccupied slot by SSs in $E_e = \{a, c, d, e\}$. Similarly, we assign slot 10 as c's part-2 slot because it is the only unoccupied slot by SSs in $E_c = \{a, b, c, d, e\}$. For a, since $E_a = \{a, c, d, e\}$, we assign slot 9 as its part-2 slot. Note that although slot 9 has already been assigned to b, it does not prevent a from using it because $b \notin E_a$. From Fig. 3, we can observe that any packet issued in part-1 slots can always be delivered to the BS within one frame time. However, a packet issued by e in its part-2 slot totally takes 12 slots to be delivered to the BS, which exceeds one frame time. Note that the above scheduling employs a proportional allocation in the sense that the bandwidth allocation for each SS is based on its own bandwidth demand, its children's demands, and the sum of all SSs' demands in the mesh network. The BS collects all of SSs' demands and allocates bandwidth to them by the ratio of their aggregated demands and $C_{\mathrm{max}}^{\mathrm{UL}}$. Since all the aggregated demands of SSs are divided by the same factor of $C_{\mathrm{max}}^{\mathrm{UL}}$, the resource is proportionally allocated to SSs. In addition, once a slot is allocated to an SS, the relaying slots are allocated to its parent SS too. Therefore, the allocation is done in an end-to-end perspective.

Theorem 1: Part-1 slots are collision free, and any packet issued in part-1 slots can be delivered to the destination station within one frame time.

Proof: We first prove that part-1 slots are collision free. For the uplink case, since $\sum_{\forall i} T_i^{\mathrm{UL}} = C_{\mathrm{total}}^{\mathrm{UL}}$, the total number of part-1 slots is $\sum_{\forall i}(T_i^{\mathrm{UL}}/C_{\mathrm{total}}^{\mathrm{UL}} \times N_{\mathrm{total}}^{\mathrm{UL}}) = N_{\mathrm{total}}^{\mathrm{UL}}$. Thus, there must be enough slots assigned to all SSs for their part-1 slots. In addition, since step 3 in the scheduling module guarantees that any two SSs will not select the same uplink slot, the part-1 slots in the uplink case are collision free. Similarly, for the downlink case, since $\sum_{\forall i}(T_i^{\mathrm{DL}}/C_{\mathrm{total}}^{\mathrm{DL}} \times N_{\mathrm{total}}^{\mathrm{DL}}) = N_{\mathrm{total}}^{\mathrm{DL}}$, it is guaranteed that there are enough slots assigned to all SSs. Again, since step 4 ensures that two SSs will not choose the same downlink slot, the part-1 slots in the downlink case are also collision free.

We then show that the latency of any packet issued in part-1 slots is bounded to one frame time. For the uplink case, we schedule SSs following the transmission order of MSH-CSCH:Request messages. Since this order is the reverse of the hop count to the BS, it is guaranteed that we always assign uplink slots of a child SS before its parent. In addition, since each SS has enough uplink slots to relay its children's packets, any packet issued in part-1 slots can be delivered to the BS within one frame time. For the downlink case, since we schedule SSs following the reverse order of the transmission order of MSH-CSCH:Request messages, we will always assign downlink slots of a parent SS before its children. Again, since each SS has enough downlink slots to relay packets from the BS, we can guarantee that any packet from the BS in part-1 slots can be delivered to the destination SS within one frame time. ■

Theorem 2: Part-2 slots are collision free.

Proof: We first prove that the part-2 slots in the uplink direction are collision free. In Section III-A, we have shown that each SS can be assigned with $T_i^{\mathrm{UL}}/C_{\mathrm{max}}^{\mathrm{UL}} \times N_{\mathrm{total}}^{\mathrm{UL}}$ slots without congesting the network. Thus, there are enough slots assigned to all SSs for their part-2 slots. In addition, step 3 in the scheduling module guarantees that any two SSs inside the interference range will not select the same slot. Thus, the part-2 slots in the uplink case are collision free. For the downlink case, since each SS can be assigned with $T_i^{\mathrm{DL}}/C_{\mathrm{max}}^{\mathrm{DL}} \times N_{\mathrm{total}}^{\mathrm{DL}}$ slots without congesting the network, there are also enough slots assigned to all SSs. Similarly, by step 4, we can ensure that two SSs inside the interference range will not choose the same slot. Thus, this theorem still holds in the downlink case. ■

Remark 1: The IEEE 802.16 mesh mode only supports time division duplex (TDD) for uplink and downlink traffics. The TDD framing is adaptive in that the bandwidths allocated to uplink and downlink traffics can vary. Unlike the PMP mode, there is no clear boundary between uplink and downlink slots in the mesh mode. In this paper, we assume that a slot will exclusively be used by only uplink or downlink throughout the whole network.

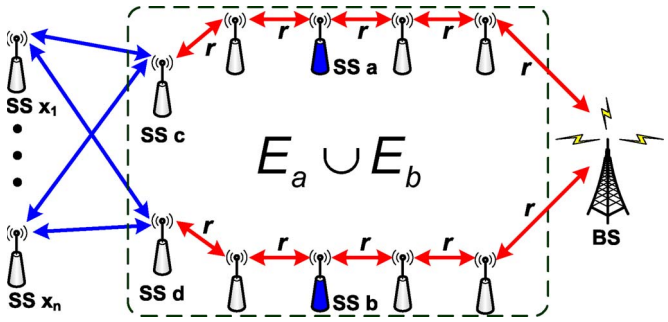


Fig. 4. Special case of the RTC problem.

C. Routing Module

In Section III-A, we have indicated that the uplink and downlink slots allocated to each SS are inversely proportional to the values of $C_{\max}^{\mathrm{UL}}$ and $C_{\max}^{\mathrm{DL}}$, respectively. Therefore, the goal of this routing module is to reconstruct the routing tree, whenever needed, to reduce both $C_{\max}^{\mathrm{UL}}$ and $C_{\max}^{\mathrm{DL}}$ so that SSs can receive more time slots.

Definition 1: Given a mesh network $\mathcal{G}$, bandwidth demands, and data rates of SSs in $\mathcal{G}$, the *RTC problem* is to find a routing tree $\mathcal{T}$ in $\mathcal{G}$ such that the value of $C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}}$ is minimized.

To prove that the RTC problem is NP-complete, we define a decision problem as follows.

Definition 2: Given a mesh network $\mathcal{G}$, bandwidth demands, data rates of SSs in $\mathcal{G}$, and a real number $\mathcal{R}$, the RTC *problem* is to decide whether $\mathcal{G}$ has a routing tree $\mathcal{T}$ such that $C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}} \leq \mathcal{R}$.

Theorem 3: The RTC problem is NP-complete.

Proof: First, given the routing trees in $\mathcal{G}$, we can calculate the values of their $C_{\max}^{\mathrm{UL}}$ and $C_{\max}^{\mathrm{DL}}$ and check whether $C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}} \leq \mathcal{R}$. Clearly, this takes polynomial time. Thus, the RTC problem belongs to NP.

We then prove that the RTC problem is NP-hard by reducing an NP-complete problem, i.e., the *partition problem* [18], to a special case of the RTC problem in polynomial time. Given a set $\mathcal{X}$ where each element $x_i \in \mathcal{X}$ has an associated size $s(x_i)$, the partition problem asks whether it can partition $\mathcal{X}$ into two subsets with equal total size.

Consider a special case of the RTC problem in Fig. 4, where the interference neighborhoods $\mathcal{I}(a)$ and $\mathcal{I}(b)$ of SS a and SS b are not overlapped. The data rates and bandwidth demands of SSs in $E_a \cup E_b$ are set to r and 0, respectively. Except for those SSs in $E_a \cup E_b$, there are n SSs $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ connected with both SS c and SS d, each with nonzero equal uplink and downlink bandwidth demands.

Here, we reduce the partition problem to the special case of the RTC problem. Let the size $s(x_i)$ be the sum of uplink and downlink bandwidth demands of each $x_i \in \mathcal{X}$, and $\mathcal{R} = 5/2 \sum_{\forall i} s(x_i)/r$. From Fig. 4, we can observe that the parent of $x_i \in \mathcal{X}$ is either SS c or SS d. Because the bandwidth demands of all SSs in $E_a \cup E_b$ are zero, the only way to make $C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}} \leq \mathcal{R}$ is to partition $\mathcal{X}$ into two subsets (where the SSs in $\mathcal{X}$ select either SS c or SS d as their parent) with equal total size. Thus, if there exists a routing tree in $\mathcal{G}$ such that $C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}} \leq \mathcal{R}$, there must be a partition to divide $\mathcal{X}$ into two subsets with equal total size. Obviously, this reduction can be performed in polynomial time. Therefore, the RTC problem is NP-complete. ■

In the following sections, we propose a heuristic *load-aware tree construction (LTC) algorithm* to solve the RTC problem. The LTC algorithm constructs the routing tree from leaves to the root. Let $P_i = P_i^{\mathrm{LS}} \cup P_i^{\mathrm{EQ}}$, where P_i^{LS} is the set of SS i's neighbors whose hop counts to the BS are less than that of SS i, and P_i^{EQ} is the set of SS i's neighbors whose hop counts to the BS are equal to that of SS i; these neighbors have already been assigned with parents. The LTC algorithm works as follows.

1) Our goal is to form a routing tree $\mathcal{T}$ to connect all SSs. Initially, SSs are not connecting to any node. Therefore, we have a forest of trees, where each tree is an individual SS. Then, we can use (1) and (2) to calculate the aggregated uplink bandwidth demand d_i^{UL}, aggregated downlink bandwidth demand d_i^{DL}, demand of uplink transmission time T_i^{UL}, and demand of downlink transmission time T_i^{DL} of each SS i. However, note that to calculate (2), it is necessary to know the parent node of SS i (to estimate the transmission rate between i and its parent). To resolve this uncertainty, we assume that before an SS i decides its actual parent, it has a *tentative parent* SS j, where $j \in P_i$, and the transmission rate between i and j is the highest among all the candidates.
2) Since the demands of the transmission times T_i^{UL} and T_i^{DL} of all nodes i are known, we can apply (3) to calculate C_i^{UL} and C_i^{DL} for all SS i.
3) Let $\mathcal{A}$ be the set of SSs that have not decided their actual parents and have maximum hop counts to the BS.
4) This step will decide the actual parent of one SS in $\mathcal{A}$.
   a) For each SS $i \in \mathcal{A}$, connect SS i to each SS $j \in P_i$ and recompute the new values of C_j^{UL} and C_j^{DL} by assuming that i's actual parent will become j. Note that to avoid forming a cycle, if the path from SS i to SS j results in a loop, we set the values of C_j^{UL} and C_j^{DL} as ∞. We then choose the SS j with the minimum value of $C_j^{\mathrm{UL}} + C_j^{\mathrm{DL}}$ as the candidate parent of SS i.
   b) Step a) will choose a candidate parent, for example, $p(i)$, for each SS $i \in \mathcal{A}$. Among these candidates, we choose the SS $p(i)$ such that the value of $C_{p(i)}^{\mathrm{UL}} + C_{p(i)}^{\mathrm{DL}}$ is minimized as the actual parent of SS i and make a connection between i and $p(i)$.
5) Repeat step 4 until the set $\mathcal{A}$ is empty.
6) Repeat steps 3–5 until all SSs have decided their actual parents.

Step 4a builds the subtree whose subtree root (SS j) has the minimum value of $C_j^{\mathrm{UL}} + C_j^{\mathrm{DL}}$. Similarly, step 4b builds the subtree whose subtree root (SS $p(i)$) has the minimum value of $C_{p(i)}^{\mathrm{UL}} + C_{p(i)}^{\mathrm{DL}}$. This can help balance the distribution of forwarding traffics and keep the final value of $C_{\max}^{\mathrm{UL}} + C_{\max}^{\mathrm{DL}}$ as small as possible in the constructed tree. Note that the above calculations of C_i^{UL} and C_i^{DL} are all tentative. Their values will keep on changing as the tree is building up. Algorithm 2 gives the pseudocode of the LTC algorithm.

Algorithm 2: *Load-aware tree construction (LTC) algorithm*

Input: set $\mathcal{G}$ of all SSs in the network
Output: routing tree $\mathcal{T}$
foreach $i \in \mathcal{G}$ **do**
  let $r^{\mathrm{UL}}_{j(\max)}$ and $r^{\mathrm{DL}}_{j(\max)}$ be the highest rates of uplinks and downlinks of SS j to SSs in P_j;
  $C^{\mathrm{UL}}_i \leftarrow \sum_{j \in E_i} \frac{b^{\mathrm{UL}}_j}{r^{\mathrm{UL}}_{j(\max)}}$;
  $C^{\mathrm{DL}}_i \leftarrow \sum_{j \in E_i} \frac{b^{\mathrm{DL}}_j}{r^{\mathrm{DL}}_{j(\max)}}$;
while $\mathcal{G} \neq \emptyset$ **do**
  let $\mathcal{A}$ be the set of SSs without parents which have the largest hop counts to the BS;
  $\mathcal{G} \leftarrow \mathcal{G} - \mathcal{A}$;
  while $\mathcal{A} \neq \emptyset$ **do**
    $C_{\min} \leftarrow \infty$;
    foreach $i \in \mathcal{A}$ **do**
      foreach $j \in P_i$ **do**
        calculate C^{UL}_j and C^{DL}_j after attaching SS i to SS j;
        if $C^{\mathrm{UL}}_j + C^{\mathrm{DL}}_j < C_{\min}$ **then**
          $C_{\min} \leftarrow C^{\mathrm{UL}}_j + C^{\mathrm{DL}}_j$;
          parent $\leftarrow j$;
          child $\leftarrow i$;
    $\mathcal{T}[\mathrm{child}] = \mathrm{parent}$;
    $\mathcal{A} \leftarrow \mathcal{A} - \{\mathrm{child}\}$;
    foreach $i \in E_{\mathrm{parent}} \cup E_{\mathrm{child}}$ **do** update C^{UL}_i and C^{DL}_i;

Next, we analyze the time complexity of the LTC algorithm. Since each SS has exactly one parent, step 4 will be repeated at most n times, where n is the number of SSs in the network. In step 4a, at most m nodes will be checked, and each will check at most d candidates, where m is the maximum number of SSs with the same hop count to the BS, and d is the maximum degree of SSs. Thus, the time complexity is $O(nmd)$.

Finally, we comment on the timing to invoke the routing module. Since reconstructing the routing tree causes communication cost, one possible moment to invoke the routing module is when the value of $C^{\mathrm{UL}}_{\max} + C^{\mathrm{DL}}_{\max}$ of the old tree is higher than that of the new tree by a predefined threshold.

IV. Bandwidth Guarantee for Real-Time Flows

The aforementioned spectral reuse framework can allocate time slots to SSs proportionate to their requests. However, when SSs request new flows or need more bandwidth for their old flows, the system may no longer guarantee enough bandwidth for the original flows. To solve this problem, we propose an *admission control* mechanism to extend our spectral reuse framework. Specifically, we separate flows into *real-time and nonreal-time* flows. When an SS requests a new flow or more bandwidth for its old flows, we will check whether the bandwidth requirements of all real-time flows can still be satisfied. If so, we will admit this request. Otherwise, we will reject this request to guarantee bandwidths of existing real-time flows.

Fig. 5 illustrates the flowchart of our admission control mechanism. The idea is to prioritize real-time from nonreal-time flows. For each SS, we always ensure sufficient slots to satisfy the bandwidth requirements of all its real-time flows, and then distribute the remaining slots to its nonreal-time flows.

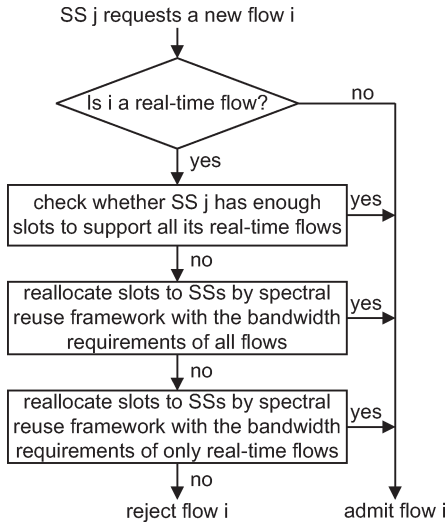


Fig. 5. Flowchart of the admission control mechanism.

This is what we mean by prioritizing real-time from nonreal-time flows. This implies that an SS can always admit more nonreal-time flows since its real-time flows always have higher priority. However, when an SS j requests a new real-time flow i (or wants to increases bandwidth of a real-time flow i), the following steps will be executed.

1) Check whether SS j's current slots can support the required bandwidths of all its real-time flows (including flow i). If there are enough slots, we can admit flow i. Otherwise, it means that we have to reallocate slots in the system to support this new request (refer to step 2).
2) To reallocate slots of SSs in the network, we will execute our spectral reuse framework in Section III. We will update the bandwidth requirement of SS j, run the routing module to reconstruct the routing tree, and then run the scheduling module to allocate slots to all SSs. Then, we check whether this new allocation can support the real-time flows of all SSs. If so, we can admit flow i and adopt the new allocation. Otherwise, it means that the new scheduling cannot satisfy some real-time flows, so we go to step 3.
3) Update the bandwidth requirements of all SSs by removing their nonreal-time flows. With these requirements, we execute our spectral reuse framework again. We run the routing module to reconstruct the routing tree and then run the scheduling module to allocate slots to all SSs. Then, we check whether this new allocation can support the real-time flows of all SSs. If so, we can admit flow i and adopt the new allocation. Otherwise, the system does not have enough slots to support flow i, so we should reject the request of flow i.

Note that although step 3 allocates slots to SSs based on their requirements of real-time flows, an SS can still transmit nonreal-time flows as long as its real-time flows do not consume all the bandwidths of the SS. In addition, we comment that although the above discussions only cover two classes (real time and nonreal-time) of traffic, general multiple m classes of traffic is applicable. In this case, we should check whether the addition of a new flow i (e.g., in class $k < m$) can still guarantee the

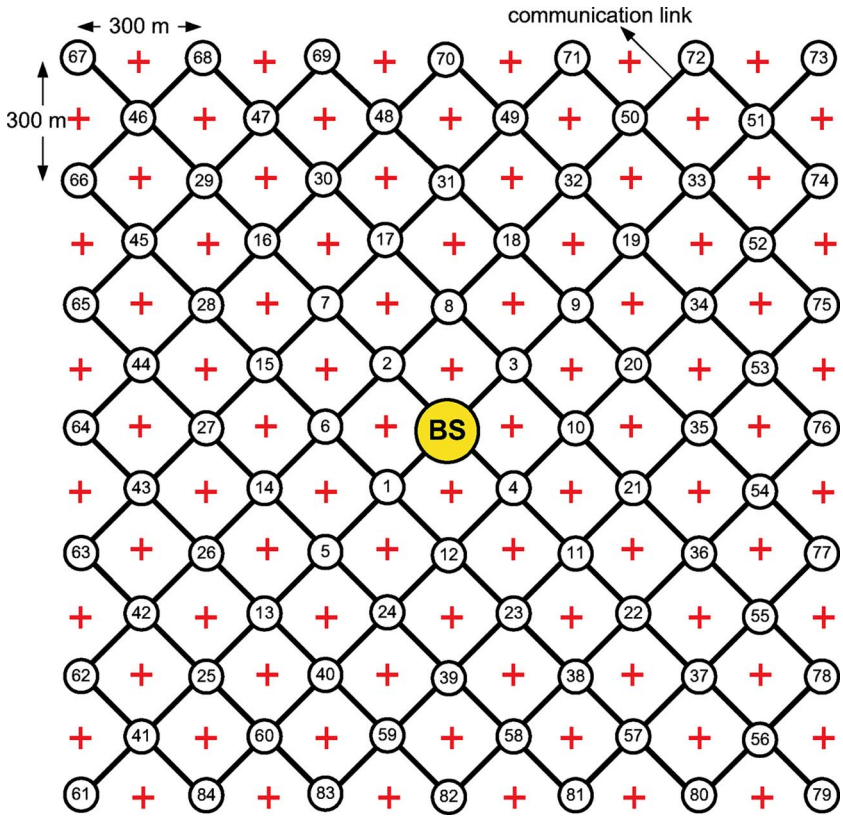


Fig. 6. Regular and dense network topologies in our experiments.

bandwidth requirements of all flows in classes $1, 2, \ldots, k$. If not, we can remove flows in classes $k+1, k+2, \ldots, m$ and reallocate slots to check whether the system has enough slots to support the request of flow i.

V. Performance Evaluation

In this section, we present some experimental results conducted by the ns-2 simulator [19] to verify the effectiveness of the proposed framework. We adopt a single-channel OFDM physical layer and a two-ray ground reflection model for radio propagation and extend the time-division multiple-access (TDMA) MAC module in ns-2 for the MAC layer. We consider three kinds of network topologies, i.e., *regular*, *dense*, and *random*. In a regular network, there are at most 84 SSs placed in a diamond mesh topology (as shown in Fig. 6). In a dense network, we add an extra SS in each position marked by "+" in Fig. 6. In a random network, we arbitrarily select at most 84 positions from the dense network to place SSs. Note that the resulting network is connected. All SSs are stationary and work in half duplex. The interference neighborhood of an SS includes all its neighbors within the two-hop range. Therefore, there are at most 12 and 24 nodes in an SS's interference range in regular and dense networks, respectively. In the random network, an SS's interference range contains 12 nodes on average. There are 512 time slots in a frame. The channel bandwidth is set to 50 Mb/s, and we assume that all links have the same data rates. For each experiment, at least 100 simulations are repeated, and we take their average.

A. Network Throughputs Under Different Network Topologies

We first evaluate the network throughputs under different network topologies. *Network throughput* is defined as the total amount of data received and transmitted at the BS. We compare our results against the basic 802.16 mesh operation and the concurrent transmission scheme with route adjustment proposed in [17]. For the 802.16 operation, the random routing tree is adopted, and the numbers of uplink and downlink slots are set to equal. Each SS will generate random traffic loads and request the same uplink and downlink bandwidth demands. For regular and random networks, the number of SSs is set to 4, 12, 24, 40, 60, and 84. For the dense network, we set the number of SSs as 8, 24, 48, 80, 120, and 168.

Fig. 7 shows the network throughputs of different methods in the regular network. Clearly, the network throughput will decrease as the number of SSs increases because a packet needs to travel more hops on average as the network scales up. From Fig. 7, we can observe that the throughput of the 802.16 operation significantly drops when the number of SSs increases. This is because it adopts a random routing tree, which causes longer relay routes. Moreover, the neglect of spectral reuse greatly hurts the system performance. The improvement of throughput by the concurrent transmission scheme proposed in [17] is limited because it constructs the routing tree according to the SSs' positions rather than their traffic loads. Thus, the network bottleneck cannot be reflected, and the benefit of route adjustment is limited. Moreover, this concurrent transmission scheme restricts that SSs cannot transmit data earlier than their

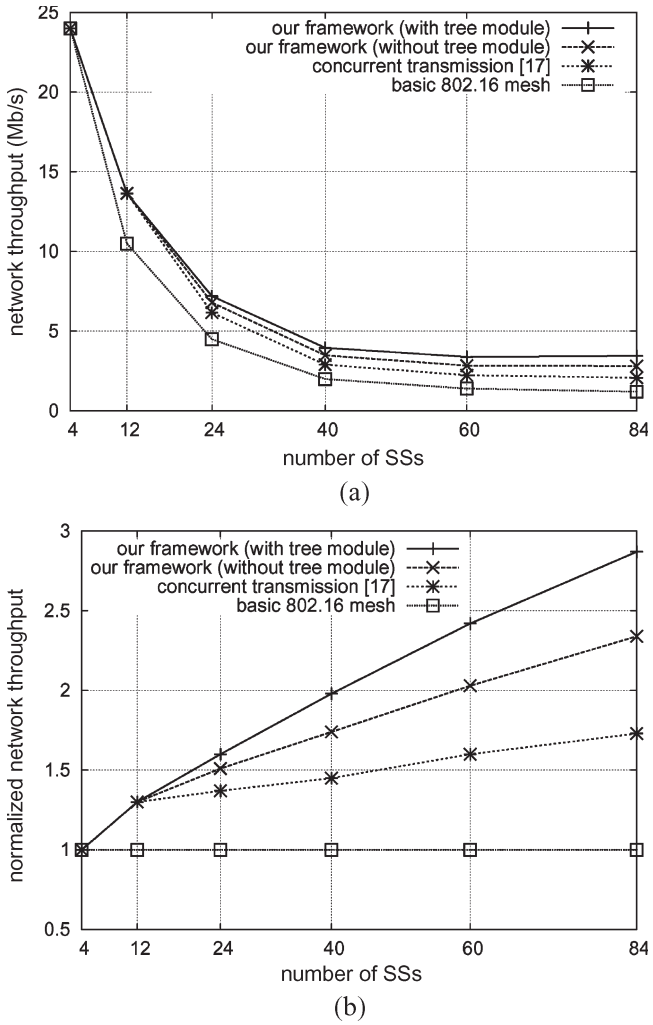


Fig. 7. Comparison of network throughputs in the regular network.

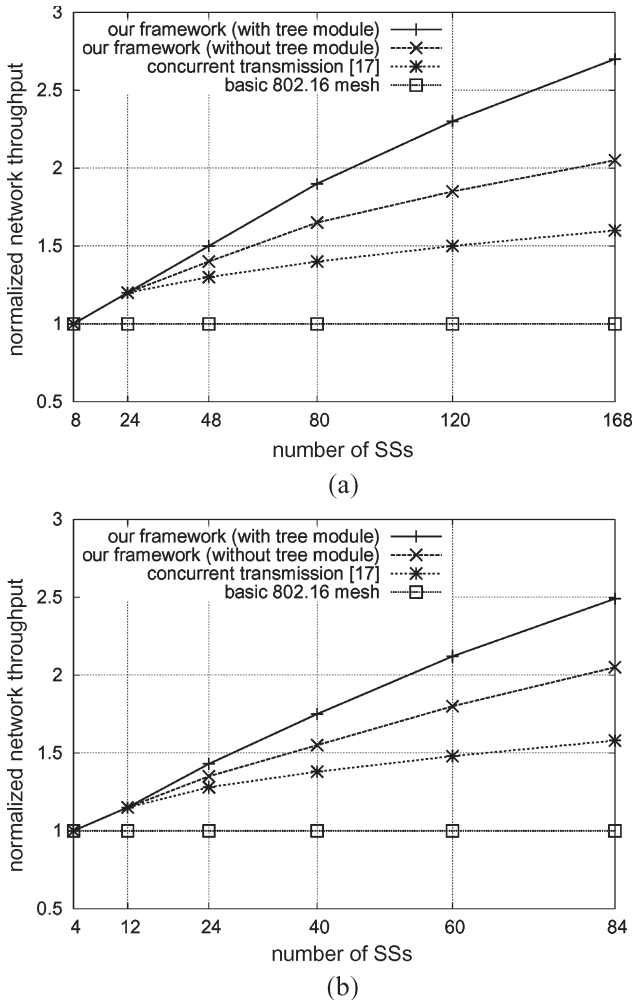


Fig. 8. Comparison of normalized network throughputs in dense and random networks.

child SSs so that the throughput is reduced. Our framework performs better than these two schemes because it can estimate the degree of spectral reuse according to SSs' traffic loads and thus allocates more time slots to SSs. As the network scale grows, the degree of spectral reuse can also increase. In addition, the LTC algorithm of the tree module can generate better routing paths to distribute the traffics more evenly. Therefore, the complete framework can result in the highest throughput.

We then verify the network throughputs of different methods in dense and random networks (as shown in Fig. 8). All the network throughputs are normalized by that of the basic 802.16 mesh operation. From Fig. 8, we can observe that the results are similar to that in Fig. 7. However, as compared with Fig. 7, the improvement of our framework slightly degrades. For the dense network, this is due to the decrease of the degree of spectral reuse since the number of nodes in each SS's interference neighborhood becomes double. For the random network, this is because the network bottleneck usually appears in the one-hop neighbors of the BS.

In the following experiments, we conduct all the simulations in the regular network.

B. Network Throughputs Under Different Traffic Demands

Fig. 9 shows the normalized network throughputs under different numbers of SSs with various uplink traffic demands. Each SS randomly requests 50%–100% uplink bandwidth demand. From Fig. 9, we can observe that the network throughput of our framework is much higher than that of the 802.16 operation. This is because the 802.16 operation only allocates equal numbers of slots to uplink and downlink traffic without any flexibility. The situation becomes worse when the number of SSs increases, because the difference between the amount of uplink traffic and the amount of downlink traffic could be large. On the contrary, our framework allocates the ratio of uplink to downlink slots as $C_{\max}^{\mathrm{UL}} : C_{\max}^{\mathrm{DL}}$, which reflects the practical traffic loads of SSs. In addition, the tree module helps reconstruct a better routing tree to reduce both values of $C_{\max}^{\mathrm{UL}}$ and $C_{\max}^{\mathrm{DL}}$, thereby further improving system performance.

Fig. 10 illustrates the normalized network throughputs under different uplink traffic demands. We set the number of SSs as 84. Each SS generates 0.3-Mb/s traffic load on average, where the ratio of uplink request is varied from 10% to 50%. From Fig. 10, we can observe that our framework can significantly improve the network throughput, particularly when the

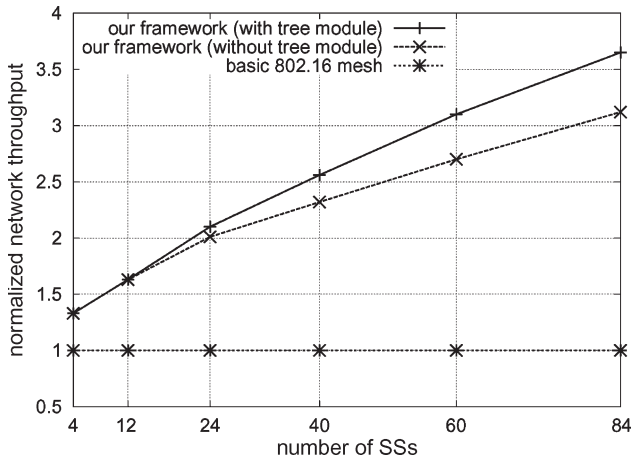


Fig. 9. Comparison of normalized network throughputs under different numbers of SSs with various uplink traffic demands.

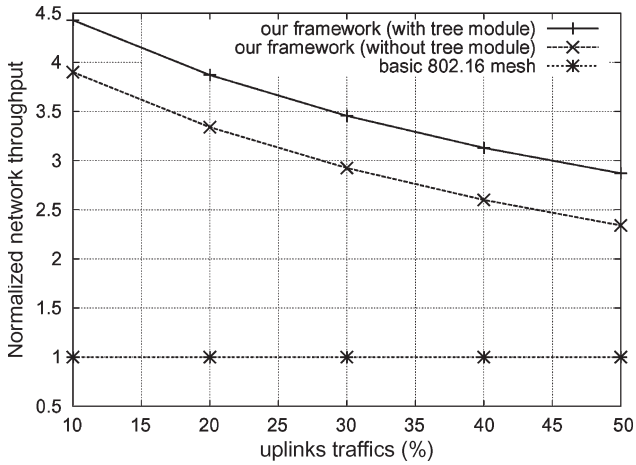


Fig. 10. Comparison of normalized network throughputs under different uplink traffic demands.

difference between uplink and downlink traffic demands increases. This is because the 802.16 operation simply allocates equal numbers of slots for uplink and downlink traffic, which may lead to network congestion in one direction while leaving slots wasted in another direction. The situation becomes worse when the traffic loads in the uplink and downlink directions become extremely unbalanced.

C. Packet Dropping Ratio of Real-Time Flows

We then evaluate the *packet dropping ratio* of real-time flows in the network, which is defined as the ratio of the number of real-time packets dropped (due to exceeding deadlines) to the number of real-time packets generated. We set the deadline of a real-time packet as 500 ms. There are 80% real-time flows and 20% nonreal-time flows in the network. Fig. 11 illustrates the packet dropping ratios under different numbers of SSs. We can observe that our framework can result in a lower packet dropping ratio because it can achieve a higher network throughput with the help of spectral reuse and tree reconstruction. Therefore, real-time flows can receive more bandwidth to alleviate their packet dropping ratios.

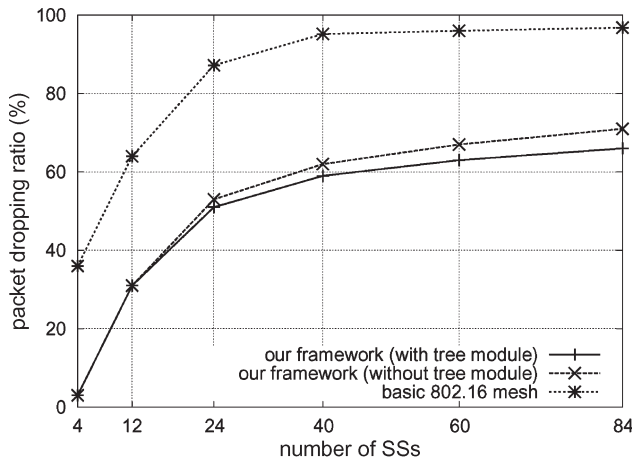


Fig. 11. Comparison of packet dropping ratios under different numbers of SSs.

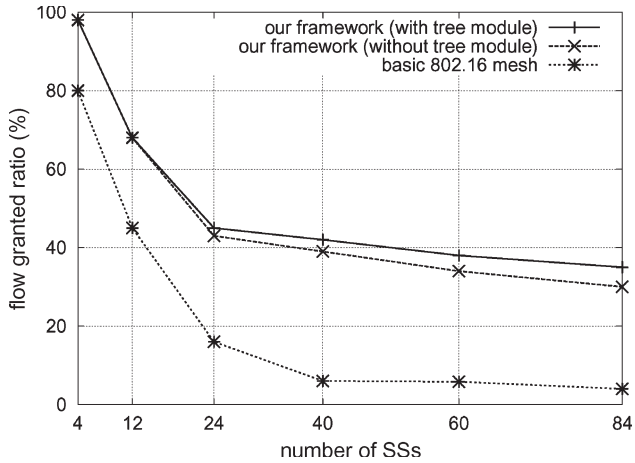


Fig. 12. Comparison of real-time-flow-granted ratios under different numbers of SSs.

D. Real-Time Flow Granted Ratio

Fig. 12 shows the real-time-flow-granted ratio under different numbers of SSs. The *real-time-flow-granted ratio* is defined as the ratio of the number of admitted real-time flows to the number of requested real-time flows. We set the ratio of the number of real-time flows to the number of nonreal-time flows as 4 : 1. Each flow uniformly generates a traffic load of [0.1 Mb/s, 0.5 Mb/s]. From Fig. 12, we can observe that when the number of SSs increases, the real-time-flow-granted ratio will decrease because the average routing path to the BS increases. In this case, SSs have to relay more traffic from their children, resulting in a high risk of network congestion. By exploiting spectral reuse, our framework can achieve a higher network throughput and thus improves the real-time-flow-granted ratio. In addition, the extension of our framework in Section IV prioritizes real-time flows from nonreal-time flows, thereby further improving the real-time-flow-granted ratio.

Fig. 13 illustrates the real-time-flow-granted ratio under different traffic loads of 84 SSs. We vary the average traffic load of SSs from 0.1 to 0.6 Mb/s. Each SS will request 80% real-time flows and 20% nonreal-time flows. From Fig. 13, we can observe that the real-time-flow-granted ratio significantly decreases as the average traffic load increases due to serious

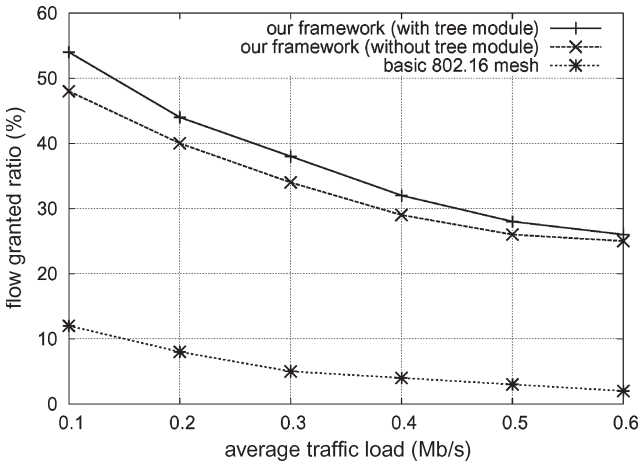


Fig. 13. Comparison of real-time-flow-granted ratios under different traffic loads.

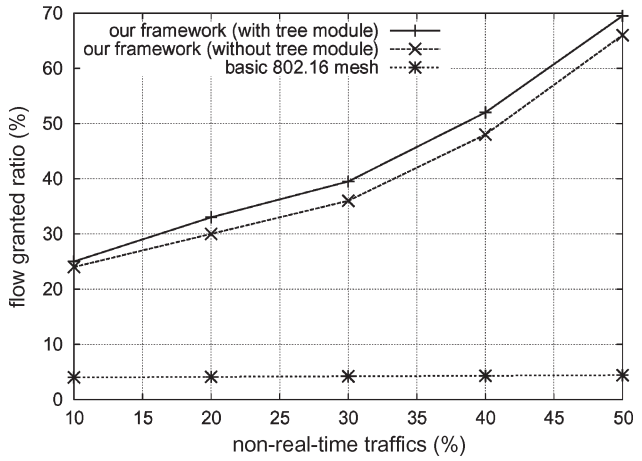


Fig. 14. Comparison of real-time-flow-granted ratios under different nonreal-time traffic demands.

network congestion. In such a severe environment, the 802.16 operation can only admit no more than 10% real-time flows. On the other hand, our framework can still admit 25% real-time flows even when the average traffic load of SSs arrives to 0.6 Mb/s. This reflects the flexibility of flow scheduling in our framework.

Fig. 14 shows the real-time-flow-granted ratio under different nonreal-time traffic demands. We set the number of SSs as 84. Each SS generates 0.3-Mb/s traffic load in average, where the ratio of nonreal-time flows is varied from 10% to 50%. From Fig. 14, we can observe that the real-time-flow-granted ratio of our framework can be improved as the ratio of nonreal-time flows increases because real-time flows can obtain more bandwidths from these nonreal-time flows.

VI. Conclusion and Future Work

In this paper, we have shown how to increase the degree of spectral reuse in an IEEE 802.16 mesh network. An integrated spectral reuse framework for centralized scheduling and an RTC scheme are developed. Compared with previous works, our framework is complete in exploiting the spectral reuse of IEEE 802.16 mesh networks in the sense that it takes dynamic traffic loads of SSs into account and integrates not only a bandwidth-scheduling scheme but also a time-slot allocation scheme. In addition, a routing algorithm with tree optimization is proposed. We have also developed an extension of our framework to support the bandwidth requirements of real-time flows. Simulation results have shown that the proposed framework significantly improves the network throughput and the flow-granted ratio compared with the specification in the IEEE 802.16 standard.

Our discussion has focused on the bandwidth guarantee of real-time flows. As for future works, several directions may deserve further investigation. First, more QoS factors of real-time flows such as delay constraints and jitters could be considered in the slot assignment strategy [20]. Second, flow differentiation rather than flow prioritization could be considered in the bandwidth allocation scheme to prevent nonreal-time flows from starvation. Third, multipath routing and distributed scheduling could be considered to provide better performance. Finally, the limitation that a slot is only exclusively used for uplink or downlink throughout the whole network could be relaxed for better bandwidth efficiency.

References

[1] *IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Std. 802.16-2004, 2004.

[2] A. Ghosh, D. R. Wolter, J. G. Andrews, and R. Chen, "Broadband wireless access with WiMax/802.16: Current performance benchmarks and future potential," *IEEE Commun. Mag.*, vol. 43, no. 2, pp. 129–136, Feb. 2005.

[3] C. Eklund, R. B. Marks, K. L. Stanwood, and S. Wang, "IEEE Standard 802.16: A technical overview of the WirelessMAN air interface for broadband wireless access," *IEEE Commun. Mag.*, vol. 40, no. 6, pp. 98–107, Jun. 2002.

[4] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: A survey," *Comput. Netw. ISDN Syst.*, vol. 47, no. 4, pp. 445–487, Mar. 2005.

[5] G. Chu, D. Wang, and S. Mei, "A QoS architecture for the MAC protocol of IEEE 802.16 BWA system," in *Proc. IEEE Int. Conf. Commun., Circuits Syst. West Sino Expo.*, 2002, pp. 435–439.

[6] M. Hawa and D. Petr, "Quality of service scheduling in cable and broadband wireless access systems," in *Proc. IEEE Int. Workshop Quality Service*, 2002, pp. 247–255.

[7] K. Wongthavarawat and A. Ganz, "IEEE 802.16 based last mile broadband wireless military networks with quality of service support," in *Proc. IEEE Military Commun. Conf.*, 2003, pp. 779–784.

[8] V. Gunasekaran and F. C. Harmantzis, "Affordable infrastructure for deploying WiMAX systems: Mesh vs. non mesh," in *Proc. IEEE Veh. Technol. Conf.*, 2005, pp. 2979–2983.

[9] M. Cao, W. Ma, Q. Zhang, X. Wang, and W. Zhu, "Modelling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode," in *Proc. ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2005, pp. 78–89.

[10] S. M. Cheng, P. Lin, D. W. Huang, and S. R. Yang, "A study on distributed/centralized scheduling for wireless mesh network," in *Proc. ACM Int. Conf. Wireless Commun. Mobile Comput.*, 2006, pp. 599–604.

[11] H. Shetiya and V. Sharma, "Algorithms for routing and centralized scheduling to provide QoS in IEEE 802.16 mesh networks," in *Proc. ACM Workshop Wireless Multimedia Netw. Performance Model.*, 2005, pp. 140–149.

[12] J. Chen, W. Jiao, and Q. Guo, "An integrated QoS control architecture for IEEE 802.16 broadband wireless access systems," in *Proc. IEEE Global Telecommun. Conf.*, 2005, pp. 3330–3335.

[13] P. Soldati, B. Johansson, and M. Johansson, "Distributed optimization of end-to-end rates and radio resources in WiMax single-carrier networks," in *Proc. IEEE Global Telecommun. Conf.*, 2006, pp. 1–6.

[14] D. Kim and A. Ganz, "Fair and efficient multihop scheduling algorithm for IEEE 802.16 BWA systems," in *Proc. IEEE Int. Conf. Broadband Netw.*, 2005, pp. 833–839.

[15] H. Y. Wei, S. Ganguly, R. Izmailov, and Z. Haas, "Interference-aware IEEE 802.16 WiMax mesh networks," in *Proc. IEEE Veh. Technol. Conf.*, 2005, pp. 3102–3106.

[16] L. Fu, Z. Cao, and P. Fan, "Spatial reuse in IEEE 802.16 based wireless mesh networks," in *Proc. IEEE Int. Symp. Commun. Inf. Technol.*, 2005, pp. 1358–1361.
[17] J. Tao, F. Liu, Z. Zeng, and Z. Lin, "Throughput enhancement in WiMax mesh networks using concurrent transmission," in *Proc. IEEE Int. Conf. Wireless Commun., Netw. Mobile Comput.*, 2005, pp. 871–874.
[18] M. Udi, *Introduction to Algorithms: A Creative Approach*. Reading, MA: Addison-Wesley, 1989.
[19] NS-2, *The Network Simulator*. [Online]. Available: http://www.isi.edu/nsnam/ns/
[20] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," in *Proc. IEEE INFOCOM*, 2007, pp. 28–36.

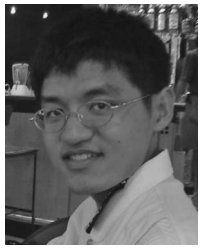
Lien-Wu Chen received the B.S. degree in computer science and information engineering from Fu Jen Catholic University, Taipei, Taiwan, in 1998 and the M.S. degree in computer science and information engineering from the National Central University, Taoyuan, Taiwan, in 2000. He is currently working toward the Ph.D. degree with the National Chiao Tung University, Hsinchu, Taiwan.

Since 2001, he has been an Engineer with Academia Sinica, Taipei. His research interests include resource management in wireless networks, link-layer protocols, and WMAN technologies.

Yu-Chee Tseng (SM'03) received the Ph.D. degree in computer and information science from The Ohio State University (OSU), Columbus, in January 1994.

He has been a Professor since 2000 and a Chairman since 2005 with the Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan, Since 2006, he has served as Adjunct Chair Professor with the Chung Yuan Christian University, Chung Li, Taiwan, His research interests include mobile computing, wireless communication, network security, and parallel and distributed computing.

Dr. Tseng received the Outstanding Research Award from the National Science Council from 2001 to 2002 and from 2003 to 2005, the Best Paper Award from the International Conference on Parallel Processing in 2003, the Elite I. T. Award in 2004, and the Distinguished Alumnus Award from OSU in 2005. He has served as Associate Editor for *Telecommunication Systems* since 2005, as Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY since 2005, and as Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING since 2006.



You-Chiun Wang (M'08) received the B.S. degree in computer science and information engineering from the National Chung-Cheng University, Chia-Yi, Taiwan, in 2001 and the M.S. degree in computer science and information engineering and the Ph.D. degree in computer science from the National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 2003 and October 2006, respectively.

He is currently a Postdoctoral Research Associate with the Department of Computer Science, National Chiao Tung University. His research interests include wireless communication, mobile computing, and sensor networks.

Da-Wei Wang (M'07) received the Ph.D. degree in computer science from Yale University, New Haven, CT.

He is currently a Research Fellow with the Institute of Information Science, Academia Sinica, Taipei, Taiwan, His research interests are in privacy enhancing technology, design, and analysis of algorithms.

Jan-Jan Wu (M'97) received the B.S. degree in computer science from the National Taiwan University, Taipei, Taiwan, in 1985 and the M.S. degree and the Ph.D. degree in computer science from Yale University, New Haven, CT, in 1991 and 1995, respectively.

She is currently an Associate Research Fellow with the Institute of Information Science, Academia Sinica, Taipei. Her research interests include parallel and distributed computing, cluster computing, and grid computing.