

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

Lien-Wu Chen, Yu-Chee Tseng, You-Chiun Wang, Da-Wei Wang, and Jan-Jan Wu

Abstract—The IEEE 802.16 standard for wireless metropolitan area networks (WMAN) is defined to meet the need of 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, bandwidth allocation, time-slot assignment, and bandwidth guarantee of real-time flows. The proposed spectral reuse framework covers bandwidth allocation at the application layer, routing tree construction and resource sharing at the MAC layer, and channel reuse at the physical layer. To the best of our knowledge, this is the first work which formally quantifies spectral reuse in IEEE 802.16 mesh networks and which 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.

1 INTRODUCTION

TO achieve the requirement of wide-range wireless broadband access at a low cost, the IEEE 802.16 standard [1] has been proposed recently. 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, DSL (digital subscriber line), or T1 links [2], [3]. The IEEE 802.16 standard is based on a common MAC (medium access control) protocol compliant with different physical layer specifications. The physical layer can employ the OFDM (orthogonal frequency division multiplexing) scheme below 11 GHz or the single carrier scheme between 10 GHz and 66 GHz.

The IEEE 802.16 MAC protocol supports the *point-to-multipoint (PMP)* mode and the *mesh* mode. In the 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 the mesh mode, stations are organized in an ad-hoc fashion. Each SS can either act as an end point or 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 wider coverage at a lower deployment cost [4].

In an IEEE 802.16 mesh network, transmissions can undergo a multi-hop 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:

- Tree reconstruction: How can we determine the routing tree based on SSs' current bandwidth demands and link qualities?
- Bandwidth allocation: How can we determine the number of time slots of each SS according to its uplink and downlink bandwidth demands?
- Time-slot assignment: How can we assign time slots to each SS in a frame?
- Bandwidth guarantee: How can 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* by another station if the latter is sufficiently separated from the former, the IEEE 802.16 standard does not explore in this direction. 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 a routing tree construction 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 to 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 non-real-time traffics so as to guarantee some bandwidths for real-time traffics. Note that the tree topology is consistent with the current IEEE 802.16 standard. Also, our framework does not require any change to

L.-W. Chen, Y.-C. Tseng, and Y.-C. Wang are with the Department of Computer Science, National Chiao-Tung University, Hsin-Chu, 30010, Taiwan. E-mail: {lwchen, yctsen, wangyc}@cs.nctu.edu.tw

D.-W. Wang and J.-J. Wu are with Institute of Information Science, Academia Sinica, Taipei, 11529, Taiwan. E-mail: {wdw, wuj}@iis.sinica.edu.tw

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 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, especially when SSs with smaller hop counts request larger bandwidths. On the contrary, our scheduling algorithm allocates time slots to SSs 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 a routing tree construction and a scheduling algorithm by considering the 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 a concurrent transmission scheme to achieve spectral reuse. As can be seen, the prior works only discuss partial aspects of the resource allocation problem.

Table 1 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 four-fold. First, it formally quantifies the spectral reuse in a mesh network, 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 also a bandwidth allocation and a time-slot assignment. Third, we propose a way to prioritize real-time from non-real-time traffics, 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, routing tree construction 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 2 briefly reviews the operations of an IEEE 802.16 mesh network and formally defines the resource allocation problem. Section 3 proposes our spectral reuse framework. Section 4 discusses how to guarantee bandwidths of real-time traffics by our framework. Section 5 gives the simulation results. Section 6 concludes this paper.

2 PRELIMINARY

2.1 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 multi-hop manner. The IEEE 802.16 MAC protocol supports both centralized and distributed scheduling methods. In this paper, we focus on the centralized scheduling to fully exploit spectral reuse.

In the centralized scheduling, the standard supports two control messages, *MSH-CSCF* (*Mesh Centralized Scheduling Configuration*) and *MSH-CSCH* (*Mesh Centralized Scheduling*), to help the BS establish its routing tree and specify 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 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. In 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 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. So the scheduling interval is about several frames depending on the size of the mesh network. Therefore, it is reasonable to assume that link data rates and bandwidth demands of SSs are constants during 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 $\frac{b_1^{UL} + b_3^{UL} + b_4^{UL}}{r_1^{UL}} : \frac{b_2^{UL}}{r_2^{UL}} : \frac{b_3^{UL}}{r_3^{UL}} : \frac{b_4^{UL}}{r_4^{UL}} (= \gamma_1 : \gamma_2 : \gamma_3 : \gamma_4)$. Note that here the calculation also includes the relay traffics. If N_{total}^{UL} is the total number of uplink slots per frame, the numbers of slots allocated to them are $\frac{\gamma_1 \cdot N_{total}^{UL}}{\sum_{i=1}^4 \gamma_i}$, $\frac{\gamma_2 \cdot N_{total}^{UL}}{\sum_{i=1}^4 \gamma_i}$, $\frac{\gamma_3 \cdot N_{total}^{UL}}{\sum_{i=1}^4 \gamma_i}$, and $\frac{\gamma_4 \cdot N_{total}^{UL}}{\sum_{i=1}^4 \gamma_i}$, respectively. The bandwidth allocation for downlink traffics 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 transmit concurrently 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

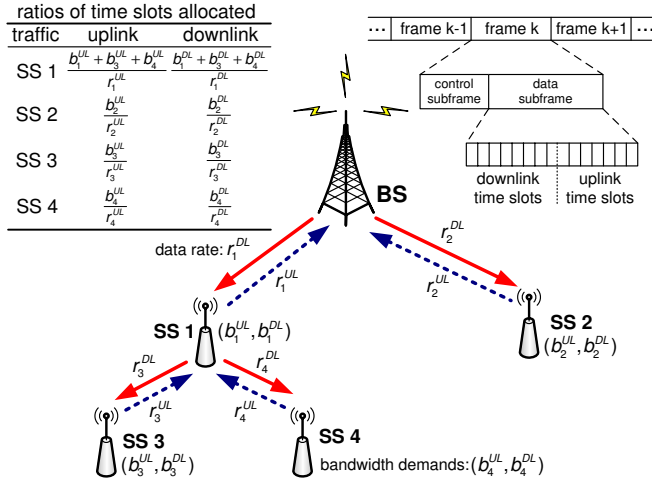
features	reuse modeling ¹	load awareness	tree reconstruction	time-slot allocation	bandwidth guarantee ³
reference [15]			partial ²	✓	
reference [16]			partial ²		
reference [17]			✓	✓	
our framework	✓	✓	✓	✓	✓

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.

TABLE 1: Comparison of prior works [15]–[17] and our spectral reuse framework.



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
$C_{\text{max}}^{\text{UL}}/C_{\text{max}}^{\text{DL}}$	maximal $C_i^{\text{UL}}/C_i^{\text{DL}}$ among all SS i in the network

TABLE 2: Summary of notations.

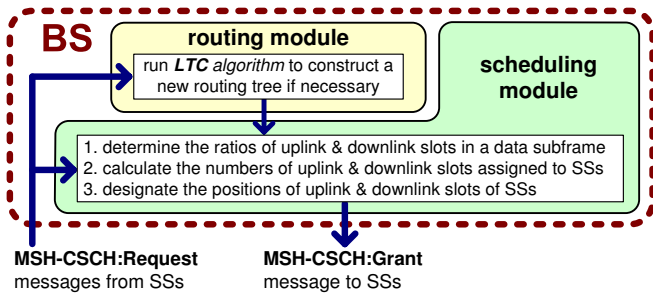


Fig. 2: System architecture of our spectral reuse framework.

proportionally. This implies that SS i can share an additional $q_i = \left(1 - \frac{C_i^{\text{UL}}}{C_{\text{total}}^{\text{UL}}}\right) \times \frac{T_i^{\text{UL}}}{C_i^{\text{UL}}}$ portion of uplink transmission time. Thus, the total portion of uplink transmission time assigned to SS i is

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

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

Unfortunately, the above Eq. (4) does not consider the congestion issue in the global network. In a non-congested network, the uplink bandwidth of an SS should be able to deliver all traffics from itself plus those from its children. Otherwise, congestion on that SS's uplink will occur. Therefore, given a non-congested 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,

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

The minimum ratio of increase among all SSs is

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

where $C_{\text{max}}^{\text{UL}} = \max_{\forall i} \{C_i^{\text{UL}}\}$. Therefore, using α_{\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

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

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

Note that the above calculation includes the demands of individual SSs as well as relay traffics. So our slot allocation is in an end-to-end sense. Next, we discuss how to adopt this concept to the scheduling module to increase 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.

3.2 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 $C_{\text{max}}^{\text{UL}} : C_{\text{max}}^{\text{DL}}$. Thus, the numbers of uplink and downlink slots in a data subframe observed by the BS are $N_{\text{total}}^{\text{UL}} = \left\lfloor \frac{C_{\text{max}}^{\text{UL}}}{C_{\text{max}}^{\text{UL}} + C_{\text{max}}^{\text{DL}}} \times N \right\rfloor$ and $N_{\text{total}}^{\text{DL}} = \left\lfloor \frac{C_{\text{max}}^{\text{DL}}}{C_{\text{max}}^{\text{UL}} + C_{\text{max}}^{\text{DL}}} \times N \right\rfloor$, respectively¹.
2. Based on $N_{\text{total}}^{\text{UL}}$ and $N_{\text{total}}^{\text{DL}}$, we then allocate $N_i^{\text{UL}} = \frac{T_i^{\text{UL}}}{C_{\text{max}}^{\text{UL}}} \times N_{\text{total}}^{\text{UL}}$ and $N_i^{\text{DL}} = \frac{T_i^{\text{DL}}}{C_{\text{max}}^{\text{DL}}} \times N_{\text{total}}^{\text{DL}}$ slots to each SS i for its uplink and downlink traffics, respectively. Note that since spectral reuse is considered, it is possible that $\sum_{\forall i} N_i^{\text{UL}} > N_{\text{total}}^{\text{UL}}$ and $\sum_{\forall i} N_i^{\text{DL}} > N_{\text{total}}^{\text{DL}}$.
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 $\frac{T_i^{\text{UL}}}{C_{\text{total}}^{\text{UL}}} \times N_{\text{total}}^{\text{UL}}$ slots. Part 2 contains $\left(\frac{T_i^{\text{UL}}}{C_{\text{max}}^{\text{UL}}} - \frac{T_i^{\text{UL}}}{C_{\text{total}}^{\text{UL}}}\right) \times N_{\text{total}}^{\text{UL}}$ 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

1. Recall that $C_{\text{max}}^{\text{UL}}$ and $C_{\text{max}}^{\text{DL}}$ represent the maximum uplink and downlink demands, respectively, seen by individual nodes. They are bottlenecks of uplink and downlink transmissions. So we use the ratio of $C_{\text{max}}^{\text{UL}}$ and $C_{\text{max}}^{\text{DL}}$ 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 $C_{\text{max}}^{\text{UL}}$ and $C_{\text{max}}^{\text{DL}}$ to improve spectral reuse. Also, 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.

with a latency no more than one frame time (the reason will be explained in Theorem 1). Now we describe how these slots are determined.

- **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 order is reverse in hop-count to the BS (that is, largest hop-count first), and is retained as 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 $\frac{T_i^{UL}}{C_{total}^{UL}} \times N_{total}^{UL}$ unoccupied slots as its part-1 slots, and then mark these slots as *occupied*. This operation is repeated until all SSs are visited.
- **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 pseudo code of the above time-slot assignment scheme.

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 $\frac{T_i^{DL}}{C_{total}^{DL}} \times N_{total}^{DL}$ slots and part 2 contains $\left(\frac{T_i^{DL}}{C_{max}^{DL}} - \frac{T_i^{DL}}{C_{total}^{DL}}\right) \times N_{total}^{DL}$ 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 SSs $a, b, c,$ and d be one slot and the demand of SS e be two slots. We assume that the interference neighborhood of an SS contains all its neighbors within two-hop range. First, part-1 slots can be assigned easily 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 takes totally 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 SSs' demands and allocates bandwidth to them by the ratio of their

aggregated demands and C_{max}^{UL} . Since all aggregated demands of SSs are divided by the same factor of C_{max}^{UL} , the resource is proportionally allocated to SSs. Also, once a slot is allocated to an SS, relaying slots are allocated to its parent SS too. Therefore, the allocation is done in an end-to-end perspective.

Algorithm 1: Time-slot assignment for uplink traffics

Input: numbers of uplink slots for SSs, $\{N_1^{UL}, N_2^{UL}, \dots, N_n^{UL}\}$
Output: result of slot assignment, $transmit[n][N_{total}^{UL}]$
 // assign part-1 slots
 let SS 1, 2, \dots , n be the transmission order of MSH-CSCH:Request messages in \mathcal{T} ;
 $free \leftarrow 1$;
for $i = 1$ to n **do**
 $allocated \leftarrow free + \frac{T_i^{UL}}{C_{total}^{UL}} \times N_{total}^{UL}$;
 for $j = free$ to $allocated$ **do** $slot[j] \leftarrow i$;
 ;
 $free \leftarrow allocated$;
 // assign part-2 slots
for $i = 1$ to n **do**
 for $j = 1$ to N_{total}^{UL} **do**
 $transmit[i][j] \leftarrow NULL$;
for $i = 1$ to n **do** // mark occupied slots of SSs
 for $j = 1$ to N_{total}^{UL} **do**
 if $slot[j] \in E_i$ **then** $transmit[i][j] \leftarrow slot[j]$;
 ;
for $i = 1$ to n **do**
 $allocated = \left(\frac{T_i^{UL}}{C_{max}^{UL}} - \frac{T_i^{UL}}{C_{total}^{UL}}\right) \times N_{total}^{UL}$;
 for $j = 1$ to N_{total}^{UL} **do**
 if $allocated > 0$ and $transmit[i][j] = NULL$ **then**
 $transmit[i][j] \leftarrow i$;
 $allocated \leftarrow allocated - 1$;
 for $k = 1$ to n **do**
 if $k \in E_i$ **then** $transmit[k][j] \leftarrow i$;
 ;
 ;
 ;
 ;

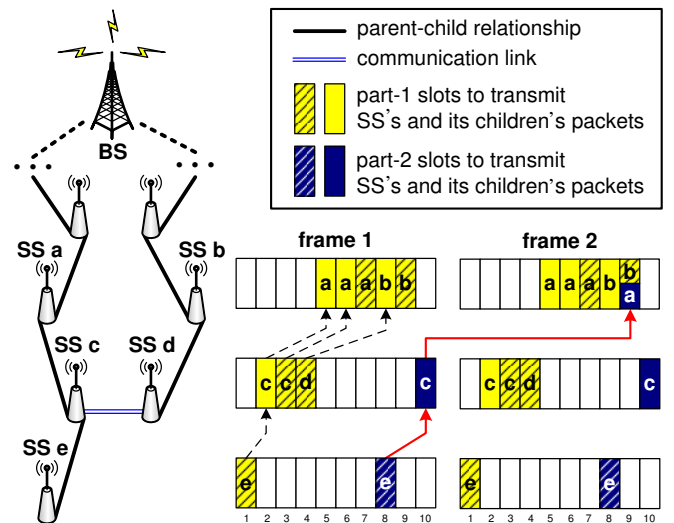


Fig. 3: An example of time-slot assignment for uplink traffics.

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 $\sum_{\forall i} T_i^{\text{UL}} = C_{\text{total}}^{\text{UL}}$, the total number of part-1 slots is $\sum_{\forall i} \left(\frac{T_i^{\text{UL}}}{C_{\text{total}}^{\text{UL}}} \times N_{\text{total}}^{\text{UL}} \right) = N_{\text{total}}^{\text{UL}}$. 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, part-1 slots in the uplink case are collision-free. Similarly, for the downlink case, since $\sum_{\forall i} \left(\frac{T_i^{\text{DL}}}{C_{\text{total}}^{\text{DL}}} \times N_{\text{total}}^{\text{DL}} \right) = N_{\text{total}}^{\text{DL}}$, 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, 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 reverse to 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. \square

Theorem 2. Part-2 slots are collision-free.

Proof: We first prove that part-2 slots in the uplink direction are collision-free. In Section 3.1, we have shown that each SS can be assigned with $\frac{T_i^{\text{UL}}}{C_{\text{max}}^{\text{UL}}} \times N_{\text{total}}^{\text{UL}}$ 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, part-2 slots in the uplink case are collision-free. For the downlink case, since each SS can be assigned with $\frac{T_i^{\text{DL}}}{C_{\text{max}}^{\text{DL}}} \times N_{\text{total}}^{\text{DL}}$ 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. \square

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 work, we assume that a slot will be used exclusively by only uplink or downlink throughout the whole network.

3.3 Routing Module

In Section 3.1, we have indicated that the uplink and downlink slots allocated to each SS is inversely proportional to the values of $C_{\text{max}}^{\text{UL}}$ and $C_{\text{max}}^{\text{DL}}$, respectively. Therefore, the goal of this routing module is to reconstruct the routing tree, whenever needed, to reduce both $C_{\text{max}}^{\text{UL}}$ and $C_{\text{max}}^{\text{DL}}$ so that SSs can receive more time slots.

Definition 1. Given a mesh network \mathcal{G} , and bandwidth demands and data rates of SSs in \mathcal{G} , the *routing tree construction (RTC) problem* is to find a routing tree \mathcal{T} in \mathcal{G} such that the value of $C_{\text{max}}^{\text{UL}} + C_{\text{max}}^{\text{DL}}$ 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 and data rates of SSs in \mathcal{G} , and a real number \mathcal{R} , the *routing tree construction (RTC) problem* is to decide whether \mathcal{G} has a routing tree \mathcal{T} such that $C_{\text{max}}^{\text{UL}} + C_{\text{max}}^{\text{DL}} \leq \mathcal{R}$.

Theorem 3. The RTC problem is NP-complete.

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

We then prove that the RTC problem is NP-hard by reducing a NP-complete problem, 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 zero, respectively. Except for those SSs in $E_a \cup E_b$, there are n SSs $\mathcal{X} = \{x_1, x_2, \dots, x_n\}$ connected with both SS c and SS d , each with non-zero equal uplink and downlink bandwidth demands.

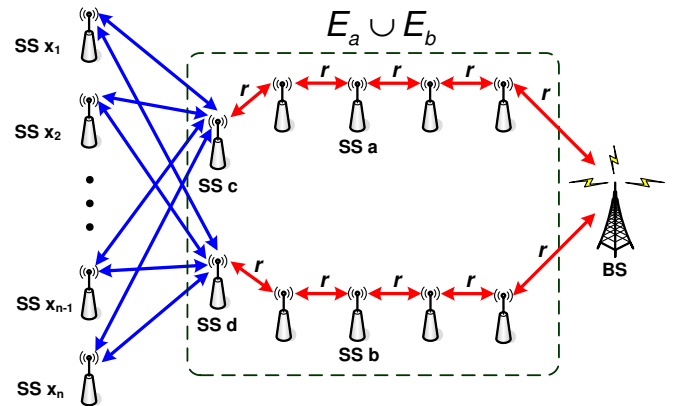


Fig. 4: A special case of the RTC problem.

Here, we reduce the partition problem to the special case of the RTC problem. Let size $s(x_i)$ be the sum of uplink and downlink bandwidth demands of each $x_i \in \mathcal{X}$, and $\mathcal{R} = \frac{5}{2} \sum_{\forall i} \frac{s(x_i)}{r}$. 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 $C_{\text{max}}^{\text{UL}} + C_{\text{max}}^{\text{DL}} \leq \mathcal{R}$ 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 $C_{\text{max}}^{\text{UL}} + C_{\text{max}}^{\text{DL}} \leq \mathcal{R}$, 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. \square

Below, 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 $P_i = P_i^{\text{LS}} \cup P_i^{\text{EQ}}$, 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 and 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. So we have a forest of trees, where each tree is an individual SS. Then we can use Eqs. (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 Eq. (2), it is necessary to know the parent node of SS i (so as 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 candidates.
2. Since the demands of transmission times T_i^{UL} and T_i^{DL} of all nodes i are known, we can apply Eq. (3) to calculate C_i^{UL} and C_i^{DL} for all SS i .
3. Let \mathcal{A} be the set of SSs which have not decided their actual parents and which have the 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 in order 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 $C_j^{\text{UL}} + C_j^{\text{DL}}$ as the candidate parent of SS i .
 - (b) The above step (a) will choose a candidate parent, say, $p(i)$ for each SS $i \in \mathcal{A}$. Among these candidates, we choose the SS $p(i)$ such that the value of $C_{p(i)}^{\text{UL}} + C_{p(i)}^{\text{DL}}$ 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, 4, and 5, until all SSs have decided their actual parents.

Step 4(a) is to build the subtree whose subtree root (SS j) has the minimum value of $C_j^{\text{UL}} + C_j^{\text{DL}}$. Similarly, step 4(b) is to build the subtree whose subtree root (SS $p(i)$) has the minimum value of $C_{p(i)}^{\text{UL}} + C_{p(i)}^{\text{DL}}$. This can help balance the distribution of forwarding traffics and keep the final value of $C_{\max}^{\text{UL}} + C_{\max}^{\text{DL}}$ 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 pseudo code of the LTC algorithm.

Next, we analyze the time complexity of the LTC algorithm. Since each SS has exact one parent, step 4 will be repeated at most n times, where n is the number of SSs in the network. In step 4(a), 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 $C_{\max}^{\text{UL}} + C_{\max}^{\text{DL}}$ of the old tree is higher than that of the new tree by a predefined threshold.

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 $r_{j(\max)}^{\text{UL}}$ and $r_{j(\max)}^{\text{DL}}$ be the highest rates of uplinks and downlinks of SS j to SSs in P_j ;

$$C_i^{\text{UL}} \leftarrow \sum_{j \in E_i} \frac{b_j^{\text{UL}}}{r_{j(\max)}^{\text{UL}}};$$

$$C_i^{\text{DL}} \leftarrow \sum_{j \in E_i} \frac{b_j^{\text{DL}}}{r_{j(\max)}^{\text{DL}}};$$

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_j^{UL} and C_j^{DL} after attaching SS i to SS j ;

if $C_j^{\text{UL}} + C_j^{\text{DL}} < C_{\min}$ **then**

$C_{\min} \leftarrow C_j^{\text{UL}} + C_j^{\text{DL}}$;

parent $\leftarrow j$;

child $\leftarrow i$;

$\mathcal{T}[\text{child}] = \text{parent}$;

$\mathcal{A} \leftarrow \mathcal{A} - \{\text{child}\}$;

foreach $i \in E_{\text{parent}} \cup E_{\text{child}}$ **do** update C_i^{UL} and C_i^{DL} ;

;

4 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 bandwidths for their old flows, the system may no longer guarantee enough bandwidths 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 *non-real-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 be still 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 non-real-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 non-real-time flows. This is what we mean by prioritizing real-time from non-real-time flows. This implies that an SS can always admit more non-real-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 increase bandwidth of a real-time flow i), the following steps will be executed:

1. Check whether SS j 's current slots can support 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).

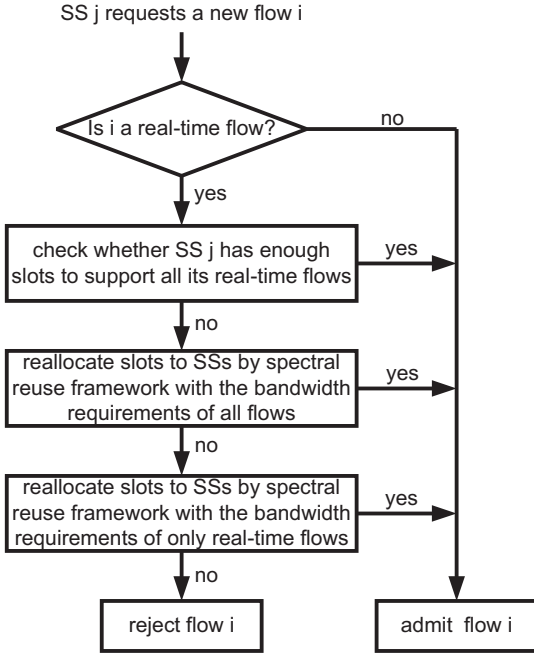


Fig. 5: Flowchart of the admission control mechanism.

2. To reallocate slots of SSs in the network, we will execute our spectral reuse framework in Section 3. 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 non-real-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 the above step 3 allocates slots to SSs based on their requirements of real-time flows, an SS can still transmit non-real-time flows, as long as its real-time flows do not consume all bandwidths of the SS. Also, we comment that although the above discussions only cover two classes (real-time and non-real-time) of traffics, general multiple m classes of traffics are applicable. In this case, we should check whether the addition of a new flow i (say, in class $k < m$) can still guarantee the bandwidth requirements of all flows in classes $1, 2, \dots, k$. If not, we can remove flows in classes $k + 1, k + 2, \dots, m$ and reallocate slots to check whether the system has enough slots to support the request of flow i .

5 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 TDMA (time division multiple access) MAC module in ns-2 for the MAC layer. We consider three kinds of network topologies: *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 two-hop range. So there are at most 12 and 24 nodes in an SS's interference range in the regular and dense networks, respectively. In the random network, an SS's interference range contains 12 nodes in 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.

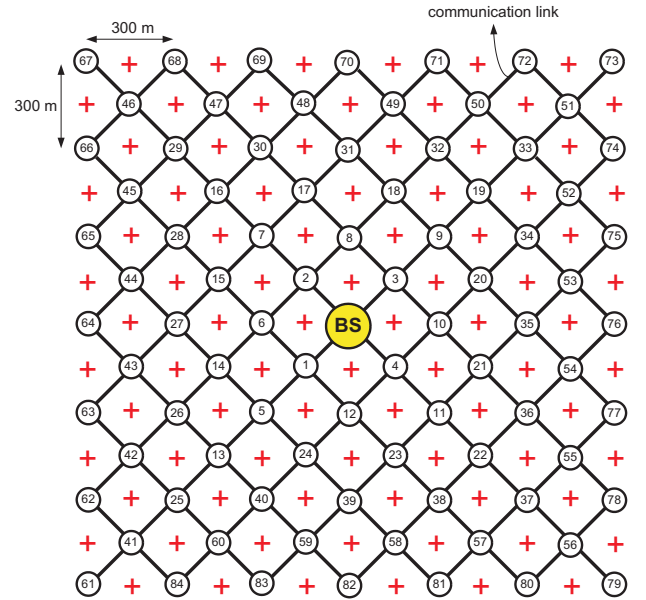


Fig. 6: The regular and dense network topologies in our experiments.

5.1 Network Throughputs under Different Network Topologies

We first evaluate the network throughputs under different network topologies. The *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 the 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 in average as the network scales

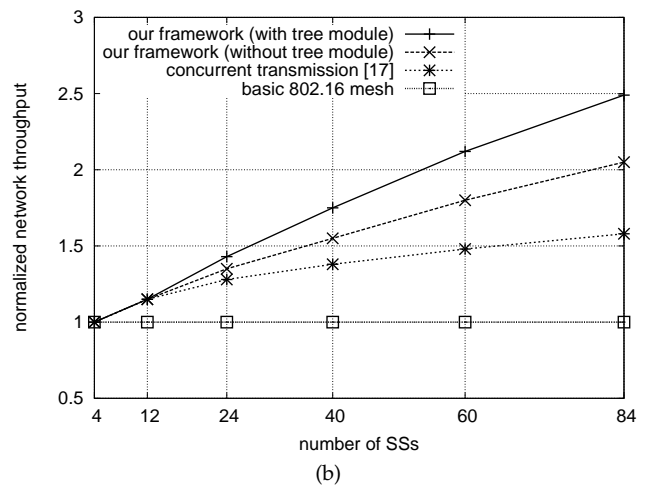
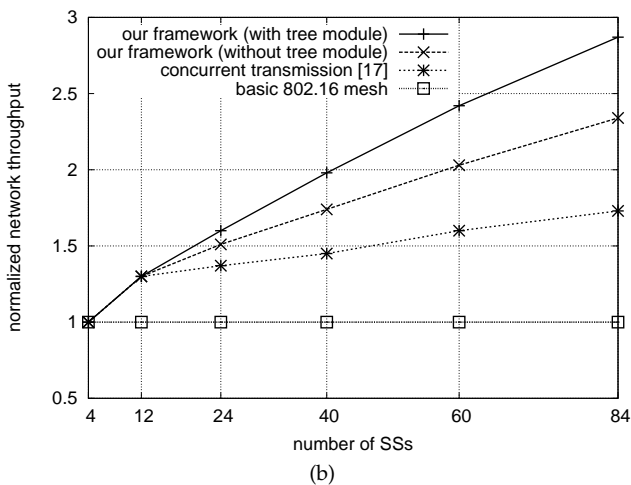
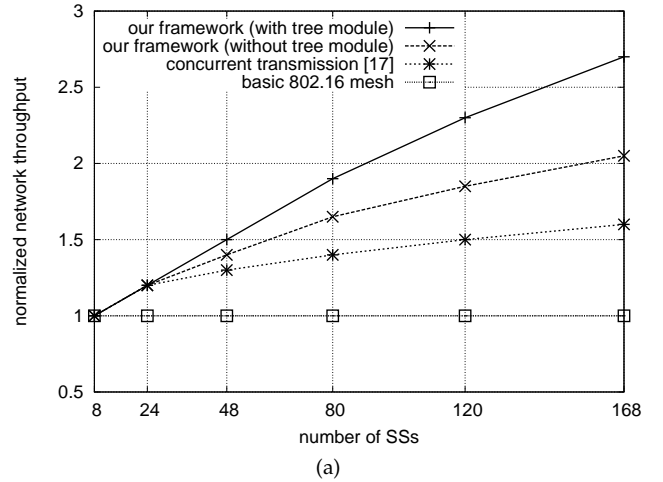
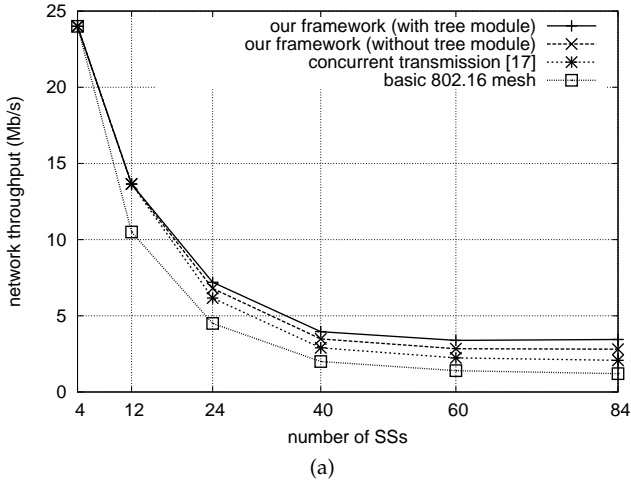


Fig. 7: Comparison of network throughputs in the regular network: (a) network throughputs and (b) normalized network throughputs.

Fig. 8: Comparison of normalized network throughputs in the dense and random networks: (a) dense network and (b) random network.

up. From Fig. 7, we can observe that the throughput of the 802.16 operation drops significantly 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. Besides, this concurrent transmission scheme restricts that SSs cannot transmit data earlier than their 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 the dense and random networks, as shown in Fig. 8. All 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 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 simulations in the regular network.

5.2 Network Throughputs under Different Traffic Demands

Fig. 9 shows the normalized network throughputs under different number of SSs with various uplink traffic demands. Each SS randomly requests 50% to 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 traffics without any flexibility. The situation becomes worse when the number of SSs increases, because the difference between the amount of uplink traffics and the amount of downlink traffics could be large. On the contrary, our framework allocates the ratio of uplink to downlink slots as $C_{\max}^{UL} : C_{\max}^{DL}$, which reflects the practical traffic loads of SSs. In addition, the tree module helps reconstruct a better routing tree to reduce both the values

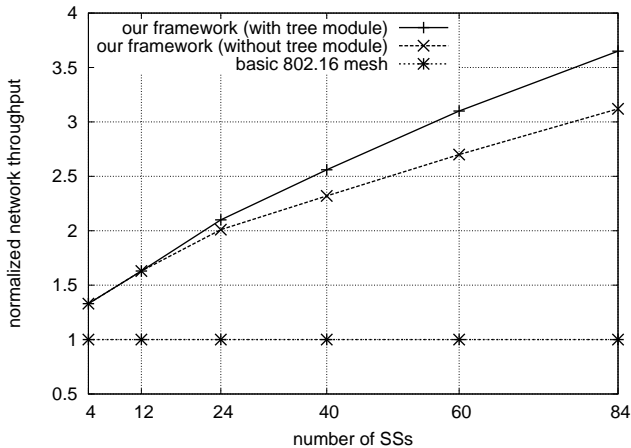


Fig. 9: Comparison of normalized network throughputs under different number of SSs with various uplink traffic demands.

of C_{\max}^{UL} and C_{\max}^{DL} , thereby further improving the system performance.

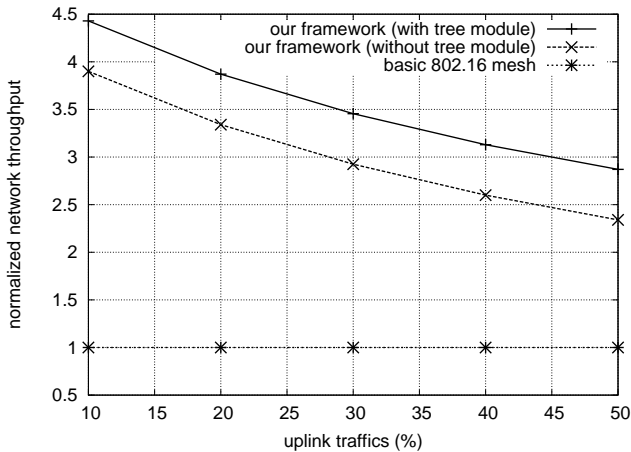


Fig. 10: Comparison of normalized network throughputs under different uplink traffic demands.

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 in 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, especially when the 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 traffics, which may lead to network congestion in one direction while leave slots wasted in another direction. The situation becomes worse when the traffic loads in uplink and downlink directions become extremely unbalanced.

5.3 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% non-real-time flows in the network. Fig. 11 illustrates the packet dropping ratios under different number

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 bandwidths to alleviate their packet dropping ratios.

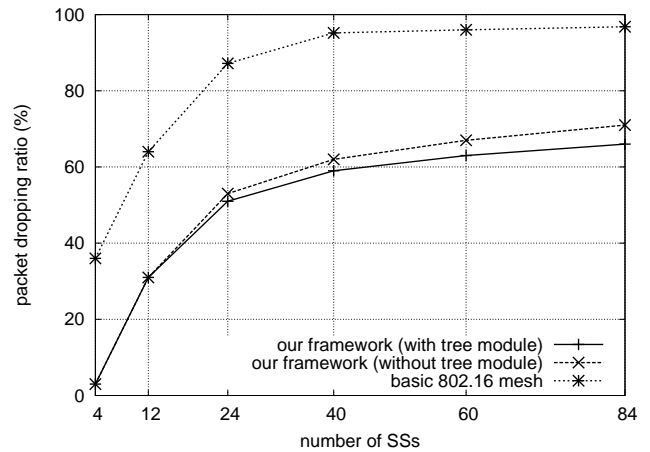


Fig. 11: Comparison of packet dropping ratios under different number of SSs.

5.4 Real-Time Flow Granted Ratio

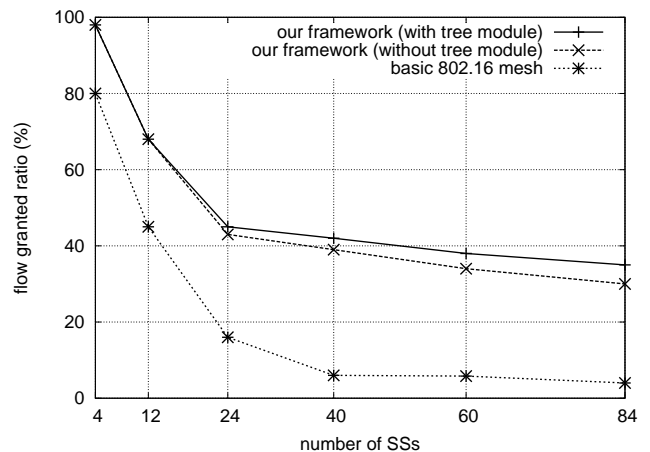


Fig. 12: Comparison of real-time flow granted ratios under different number of SSs.

Fig. 12 shows the real-time flow granted ratio under different number 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 non-real-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 traffics 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. Besides, the extension of our framework in Section 4 prioritizes real-time from non-real-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

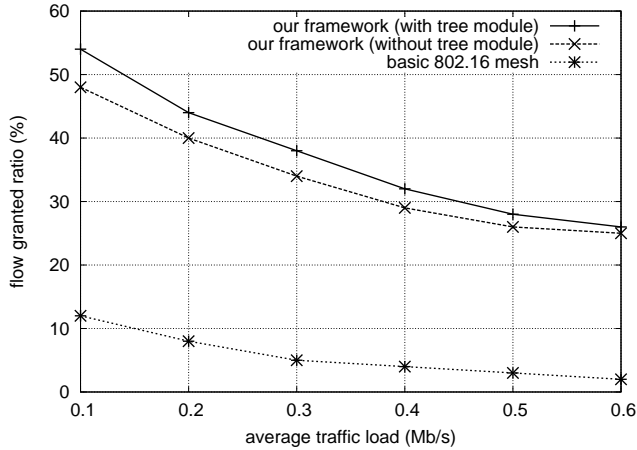


Fig. 13: Comparison of real-time flow granted ratios under different traffic loads.

load of SSs from 0.1 Mb/s to 0.6 Mb/s. Each SS will request 80% real-time flows and 20% non-real-time flows. From Fig. 13, we can observe that the real-time flow granted ratio decreases significantly as the average traffic load increases due to the serious 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 the flow scheduling in our framework.

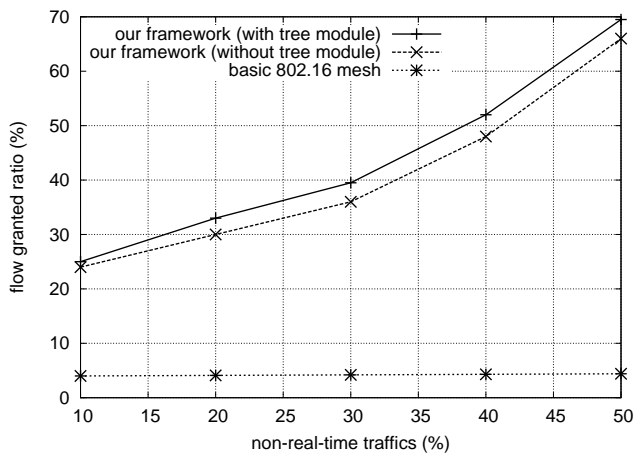


Fig. 14: Comparison of real-time flow granted ratios under different non-real-time traffic demands.

Fig. 14 shows the real-time flow granted ratio under different non-real-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 non-real-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 non-real-time flows increases because real-time flows can obtain more bandwidths from these non-real-time flows.

6 CONCLUSIONS AND FUTURE WORKS

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 a routing tree construction scheme are developed. Compared to

previous works, our framework is most complete in exploiting 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 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 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 non-real-time flows from starvation. Third, multi-path 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 802.16-2004, IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, 2004, IEEE.
- [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 Comm. Magazine*, vol. 43, no. 2, pp. 129–136, 2005.
- [3] C. Eklund, R.B. Marks, K.L. Stanwood, and S. Wang, "IEEE Standard 802.16: a technical overview of the WirelessMANTM air interface for broadband wireless access," *IEEE Comm. Magazine*, vol. 40, no. 6, pp. 98–107, 2002.
- [4] I.F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer Networks and ISDN Systems*, vol. 47, no. 4, pp. 445–487, 2005.
- [5] G. Chu, D. Wang, and S. Mei, "A QoS architecture for the MAC protocol of IEEE 802.16 BWA system," *Proc. IEEE Int'l Conf. Comm., Circuits and Systems and West Sino Expositions*, pp. 435–439, 2002.
- [6] M. Hawa and D. Petr, "Quality of service scheduling in cable and broadband wireless access systems," *Proc. IEEE Int'l Workshop Quality of Service*, pp. 247–255, 2002.
- [7] K. Wongthavarawat and A. Ganz, "IEEE 802.16 based last mile broadband wireless military networks with quality of service support," *Proc. IEEE Military Comm. Conf.*, pp. 779–784, 2003.
- [8] V. Gunasekaran and F.C. Harmantzis, "Affordable infrastructure for deploying WiMAX systems: Mesh v. non mesh," *Proc. IEEE Vehicular Technology Conf.*, pp. 2979–2983, 2005.
- [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," *Proc. ACM Int'l Symp. Mobile Ad Hoc Networking and Computing*, pp. 78–89, 2005.
- [10] S.M. Cheng, P. Lin, D.W. Huang, and S.R. Yang, "A study on distributed/centralized scheduling for wireless mesh network," *Proc. ACM Int'l Conf. Wireless Comm. and Mobile Computing*, pp. 599–604, 2006.
- [11] H. Shetiya and V. Sharma, "Algorithms for routing and centralized scheduling to provide QoS in IEEE 802.16 mesh networks," *Proc. ACM Workshop Wireless Multimedia Networking and Performance Modeling*, pp. 140–149, 2005.
- [12] J. Chen, W. Jiao, and Q. Guo, "An integrated QoS control architecture for IEEE 802.16 broadband wireless access systems," *Proc. IEEE Global Telecomm. Conf.*, pp. 3330–3335, 2005.
- [13] P. Soldati, B. Johansson, and M. Johansson, "Distributed optimization of end-to-end rates and radio resources in WiMax single-carrier networks," *Proc. IEEE Global Telecomm. Conf.*, 2006.
- [14] D. Kim and A. Ganz, "Fair and efficient multihop scheduling algorithm for IEEE 802.16 BWA systems," *Proc. IEEE Int'l Conf. Broadband Networks*, pp. 833–839, 2005.

- [15] H.Y. Wei, S. Ganguly, R. Izmailov, and Z. Haas, "Interference-aware IEEE 802.16 WiMax mesh networks," *Proc. IEEE Vehicular Technology Conf.*, pp. 3102–3106, 2005.
- [16] L. Fu, Z. Cao, and P. Fan, "Spatial reuse in IEEE 802.16 based wireless mesh networks," *Proc. IEEE Int'l Symp. Comm. and Information Technology*, pp. 1358–1361, 2005.
- [17] J. Tao, F. Liu, Z. Zeng, and Z. Lin, "Throughput enhancement in WiMax mesh networks using concurrent transmission," *Proc. IEEE Int'l Conf. Wireless Comm., Networking and Mobile Computing*, pp. 871–874, 2005.
- [18] M. Udi, *Introduction to Algorithms: A Creative Approach*, Addison-Wesley Publishing Company, 1989.
- [19] NS-2, "The network simulator," <http://www.isi.edu/nsnam/ns/>.
- [20] P. Djukic and S. Valaee, "Link scheduling for minimum delay in spatial re-use TDMA," *Proc. IEEE INFOCOM*, pp. 28–36, 2007.