Efficient Routing and Centralized Scheduling Algorithms for IEEE 802.16 Mesh Networks*

Shou-Chih Lo and Lyu-Chen Ou
Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien 974, Taiwan

Email: sclo@mail.ndhu.edu.tw

Tel: +886-3-8634029, Fax: +886-3-8634010

An IEEE 802.16 wireless system can provide broadband wireless access to subscriber stations and operate in mesh mode. The communication between a subscriber station and a base station can pass through one or more intermediate subscriber stations. The IEEE 802.16 standard provides a centralized scheduling mechanism that supports contention-free and resource-guarantee transmission services in mesh mode. However, the corresponding algorithm to this schedule is quite primitive in the standard. In this paper, we propose a more efficient way to realize this schedule by maximizing channel utilization. Our designs are divided into two phases: routing and scheduling. First, a routing tree topology is constructed from a given mesh topology by our proposed tree construction algorithm. Secondly, we allocate channel resource to the edges in the routing tree by our proposed scheduling algorithm. To further support quality-of-service schedule, we extend our designs by concerning about some issues such as service class, admission control, and fairness. Simulation results show the superiority of our proposed algorithms over others.

Keywords: IEEE 802.16, Mesh Networks, Centralized Scheduling, Routing Tree

1. INTRODUCTION

The IEEE 802.16 [1] is a standard for Wireless Metropolitan Area Networks (WMAN) and is considered as a possible substitute for DSL (Digital Subscriber Line) access technology. IEEE

* The preliminary version of this paper is published in IEEE International Conference on Scalable Computing and Communications (Scalcom), September 2009.

802.16 networks provide fixed broadband wireless access with the same level of Quality-of-Service (QoS) as traditional cabled access networks. Also, these kinds of networks provide a cheaper and more ubiquitous solution for connecting home and business to Internet.

An IEEE 802.16 system has two basic operation modes: Point-to-MultiPoint (PMP) and mesh. In PMP mode, a Subscriber Station (SS) directly communicates with the Base Station (BS). In mesh mode, an SS can indirectly communicate with the BS via other SSs. A mesh network with ad hoc deployments of SSs has the following properties [2]: communication within a local community, capacity increase, alternative routing paths, cost-effective range extension, and cost-effective deployment.

To enable data delivery with QoS guarantee in mesh networks, the IEEE 802.16 standard provides two packet scheduling schemes in centralized and distributed manners. In other words, communications among SSs can be coordinated by a centralized (and collision-free) manner over the supervision of the BS or by a distributed (and near-collision-free) manner over the mutual-adjustment of neighboring SSs. The centralized and distributed schedules, which can work together, are mainly used for Internet traffic (most traffic passes through the BS into the Internet) and Intranet traffic (most traffic results from data exchanges among SSs), respectively.

A centralized scheduling scheme with resource-guarantee transmission services is more reliable in serving traffic flows with different QoS requirements. The BS in this scheme collects bandwidth requests from the SSs and responds to them with resource grants. Then each SS performs the same scheduling algorithm to determine channel resource allocation. To achieve this, the BS splits the mesh-connected SSs up into a routing tree structure which is then used to allocate bandwidth to uplink (to the BS) and downlink (away from the BS) traffic flows.

To increase channel utilization in mesh mode, efficient routing tree construction and channel resource allocation algorithms are necessary. However, existing solutions [3] are not flexible due to some constraints such as all transmission links having the same data rate and schedule for uplink traffic only. In this paper, we provide more flexible and efficient solutions to these problems.

The remainder of this paper is organized as follows. The related work and background knowledge are given in section 2. The proposed algorithms are presented in section 3. We evaluate performance in section 4, and draw conclusions in section 5.

2. BACKGROUND KNOWLEDGE

2.1. Mesh Frame Structure

Based on Time Division Multiple Access (TDMA) technology, a channel in an IEEE 802.16 system is organized into a frame structure. If the Orthogonal Frequency Division Multiplexing (OFDM) technique is used at physical layer, each frame is consisted of some OFDM symbols each of which has a fixed time span. The number of data bits that can be carried by an OFDM symbol is determined by the modulation type (or data rate) at physical layer. A mesh frame is further divided into a control subframe and a data subframe as shown in Fig. 1. Control packets for network configuration and traffic schedule are exchanged in control subframes; however, data packets are delivered within data subframes. The length of a control subframe is fixed with MSH-CTRL-LEN minislots, where MSH-CTRL-LEN is an integer number specified by the network operator. Each control minislot is 7 OFDM symbols long. A data subframe has totally 256 minislots each of which has the following number of OFDM symbols:

$$\lceil (\text{total OFDM symbols per frame} - \text{MSH-CTRL-LEN} \times 7)/256 \rceil .$$

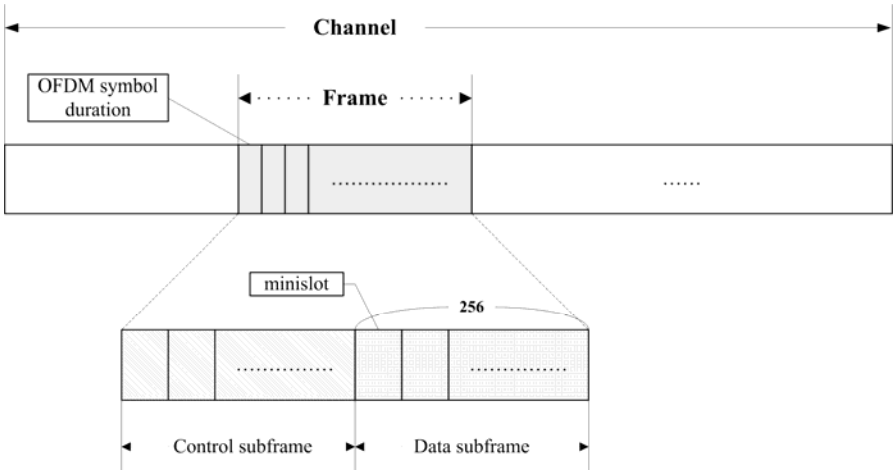


Fig. 1. Mesh frame structure.

Any packet scheduling scheme is concerned about the minislot allocation in a data subframe. If both centralized and distributed scheduling schemes work together, the centralized scheduling scheme is responsible to manage the first MSH-CSCH-DATA-FRACTION percentage of minislots in a data subframe, where MSH-CSCH-DATA-FRACTION is a pa-

rameter specified by the network operator.

2.2. Centralized Scheduling Scheme

We introduce the basic operations of a centralized scheduling scheme mentioned in the IEEE 802.16 standard. The BS first computes a breadth-first-topology-based routing tree, which is constructed by using the breadth-first traversal on a mesh topology, for the SSs that are located within a predefined hop distance from the BS. Then the BS announces the tree structure to all SSs by sending an MSH-CSCF (Mesh Centralized Schedule Configuration) control packet. The following tree information is included in this control packet: the number of nodes attached to the BS and for each of those nodes its ID and the IDs of all its child nodes, and so on and so forth, and the burst profiles (modulation type, preamble length, etc.) of all links on the tree for both uplink and downlink directions. This control packet is passed to the SSs according to the sequence of their IDs. Fig. 2a is an example routing tree, where b_{ij} is the burst profile of the link from node with ID i to node with ID j. The ID sequence is given by the breadth-first traversal order. The corresponding message to describe this tree is given in Fig. 2b.

After having the tree structure, which can be modified later if necessary, the BS collects bandwidth requests from SSs and distributes bandwidth grants to SSs. Suppose that each SS_i in the example tree has a bandwidth requirement of r_{ui} and r_{di} for uplink and downlink traffic, respectively. Then each SS would send its bandwidth request by using an MSH-CSCH (Mesh Centralized Scheduling) control packet to the BS in reversed ID order (i.e., 4, 3, 2, and 1 in Fig. 2a). A parent node aggregates bandwidth requests from its immediate children and itself into one bandwidth request. After receiving all bandwidth requests, the BS announces bandwidth grants by using the same MSH-CSCH control packet to the SSs. The bandwidth grants for our example tree are shown in Fig. 2c, where g_{ui} and g_{di} are bandwidths granted to requests r_{ui} and r_{di} respectively.

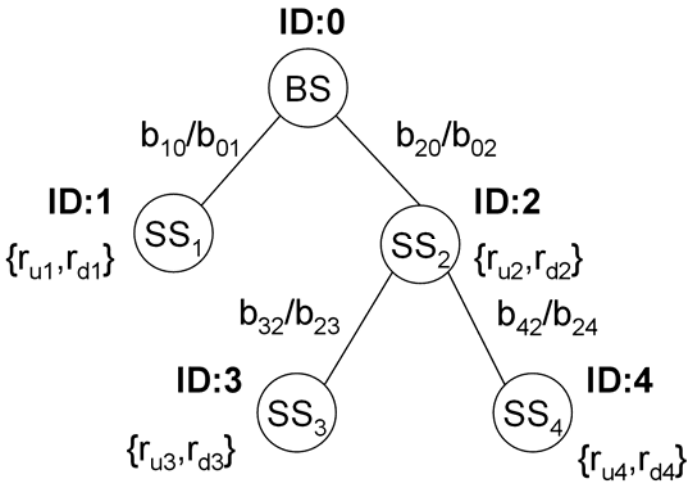


Fig. 2a. An example routing tree.

Number of nodes	5
BS	0: {(1, b_{10}, b_{01}), (2, b_{20}, b_{02})}
SS_1	1: {}
SS_2	2: {(3, b_{32}, b_{23}), (4, b_{24}, b_{42})}
SS_3	3: {}
SS_4	4: {}

Fig. 2b. Tree description message.

Link	Uplink granted bandwidth	Downlink granted bandwidth
(BS,SS_1)	g_{u1}	g_{d1}
(BS,SS_2)	$g_{u2}+ g_{u3}+ g_{u4}$	$g_{d2}+ g_{d3}+ g_{d4}$
(SS_2,SS_3)	g_{u3}	g_{d3}
(SS_2,SS_4)	g_{u4}	g_{d4}

Fig. 2c. Bandwidth grant information.

The granted bandwidth to each link is translated into the number of minislots that are actually required in each data subframe. If the total number of required minislots exceeds the upper bound, each SS reduces its requirement proportionally. Finally, each SS in the routing tree gets a global minislot allocation by executing the same scheduling algorithm. The minislot allocation guarantees that those packets sent by SSs can reach the BS within one data subframe time. For example, a data packet sent by SS_3 in Fig. 2a needs two minis-

lots in a data subframe for transmission on links (SS_3, SS_2) and (SS_2, BS). Hence, the end-to-end delay for a packet to be received by the BS is bound within one frame time. Each SS follows this minislot allocation during a schedule period that may span several frame times. A schedule period is a time interval with length sufficient for aggregating and granting all bandwidth requests in a routing tree. An SS can change its bandwidth requirement per schedule period. Unfortunately, the corresponding scheduling algorithm is not specified in the standard. To have good resource utilization, the minislot allocation should be minimized.

2.3. Related Work

Several centralized scheduling algorithms have been proposed in past years. Two main design issues have to be considered, which include routing tree construction and minislot allocation. The first issue aims to find an efficient routing tree from a mesh topology. Two common criteria are usually taken into account on routing tree construction. First, each SS in the routing tree is connected to the BS via the shortest and less interfered route path. Second, there is no single SS that experiences many aggregated traffic. The breadth-first-topology-based tree used in the IEEE 802.16 standard is based on the shortest paths. In [4], an interference-aware routing tree is introduced by connecting each SS to the BS via a route path that has the minimum degree of interference. The interference degree of a node is represented by the number of direct neighboring nodes of this node. The interference degree of a route path is then the sum of the interference degrees of nodes on this path. In [5], two tree construction algorithms: min max degree and maximum parallelism are proposed. The former one aims to minimize the maximal node degree in a routing tree, and in other words to reduce aggregated traffic to a single node. The latter one aims to maximize the number of link pairs in a routing tree that can work simultaneously without interference. The maximum parallelism routing algorithm performs best among these approaches.

The second issue aims to minimize the number of allocated minislots to the SSs in a data frame. Basically, this minislot number is fixed if minislot usage is mutual exclusive among SSs. However, we can reduce the demand for minislots by the characteristics of space diversity. Space diversity enables minislot reuse by which two transmission links can use the same minislot if they are spatially disjointed in their communication ranges (i.e., without

interference). The proposed approaches in [6][7][8] do not consider minislot reuse. The interference-aware schedule proposed in [4] allocates first one minislot to the link with the maximal bandwidth demand. Meanwhile, this same minislot will be allocated to the other links that can transmit simultaneously without interference by scanning the remaining links on the routing tree in descending order of their bandwidth demands. After updating new bandwidth demands, this process is repeated again. Three other scheduling methods, which follow a similar process with the interfere-aware one, are proposed in [9] by using the minimum interference, nearest, and farthest principles. The minimum interference method schedules first the link with the minimum interference from neighboring links. The nearest and farthest methods schedule first the link that is away from the BS with the shortest and longest distances in hops respectively. The nearest method has more efficient performance than the other two methods.

The greedy schedule proposed in [10] allocates minislots to SSs according to the sequence of their IDs. In each SS's turn, the required minislots are allocated once for all pending data packets. In [11][12], a series of scheduling algorithms is discussed, but all of them suffer from an *ordering delay* problem [3] in which an SS is scheduled to forward packets before receiving them. The ordering delay problem creates the waste of minislots. The line schedule proposed in [5] takes advantage of an optimal scheduling algorithm on a chain topology and can avoid the ordering delay problem. The line scheduling algorithm performs best among all.

More advanced issues including the usage of multiple radio interfaces and multiple channels are discussed in [13][14][15][16][17[18]. These research results are relevant to the channel assignment problem in a general mesh network, but are beyond the scope of this paper which focuses on the IEEE 802.16 architecture.

3. PROPOSED SOLUTIONS

We first formulate the centralized scheduling problem considered in the paper and then illustrate our proposed solutions. We make minislot allocation in the schedule to be coinci-

dent with the forwarding progress of PDUs (Protocol Data Units) along a routing tree. This could naturally alleviate the ordering delay problem.

3.1. Problem Formulation

We are given a topology $G = (V, E)$ of a mesh network. V is the set of nodes including the BS and the associated SSs. Let v_0 be the BS and v_i ($i > 0$) be the ith SS. The node identifier of v_i is i and is given according to the breadth-first traversal order of G. E is the set of directed links among the nodes in V. There are two directed links (forward and backward) between two nodes if they are within the transmission range of each other. e_{ij} denotes a directed link from v_i to v_j. The physical burst profile of e_{ij} is represented by b_{ij}. e_{ij} and e_{ji} might have different burst profiles and hence are not symmetric.

The granted bandwidths to node v_i ($i > 0$) for uplink and downlink traffic are g_{ui} and g_{di} (non-negative values) in bits per second respectively. These granted bandwidths can be translated into the expected number of PDUs that will be issued from or destined to v_i during a frame time according to the size of a PDU and the frame rate. Let the translation function be N_{PDU}. We can conclude that v_i will upload $N_{\mathrm{PDU}}(g_{ui})$ PDUs to the BS and download $N_{\mathrm{PDU}}(g_{di})$ PDUs from the BS. The average size of an uplink PDU and the average size of a downlink PDU are denoted by $L_{\mathrm{PDU}}(g_{ui})$ and $L_{\mathrm{PDU}}(g_{di})$ respectively. The same PDU will be transmitted using a different number of minislots on different links with various burst profiles. Let $m(L, b_{ij})$ denote the number of minislots required for transmitting a PDU of size L bytes on a link with burst file b_{ij}.

Let us denote the two route paths selected by v_i for uplink and downlink traffic in G by P_{ui} and P_{di}, respectively. A route path is represented by a set of successive links. Whether P_{ui} is equal to P_{di} is dependent on the path selection criterion. Merging all these uplink (or downlink) route paths for nodes in G constitutes an uplink (or downlink) routing tree. G_{uR} and G_{dR} respectively denote an uplink routing tree and a downlink routing tree. For successive forwards of PDUs along a route path, each link involved in this path consumes certain minislot space. The expected number of required minislots of v_i ($REQ(v_i)$) within a data subframe is represented by (1).

$$REQ(v_i) = \sum_{\forall e_{ij} \in P_{ui}} N_{PDU}(g_{ui}) \cdot m(L_{PDU}(g_{ui}), b_{ij}) + \sum_{\forall e_{ij} \in P_{di}} N_{PDU}(g_{di}) \cdot m(L_{PDU}(g_{di}), b_{ij}) \quad (1)$$

The total expected minislot requirement for all nodes in G ($REQ(G)$) is then represented by (2).

$$REQ(G) = \sum_{\forall v_i \in V} REQ(v_i) \quad (2)$$

However, the actual requirement would be smaller than this expected number if space reuse is applied. Let the minislots that are actually allocated to v_i in a data subframe be represented by set S_i, where $S_i \subset S$ and S is the set of all minislots that are available to centralized schedule in a data subframe. Therefore, the overall minislot allocation to all nodes in G is $\bigcup_{v_i \in V} S_i$. An overall minislot allocation is said to be valid if the following condition is always true: If there are any two nodes v_i and v_j in G and $S_i \cap S_j \neq \phi$, the corresponding two transmission links to the same minislot should not be mutually interfered.

The conditions are defined below when two links e_{AB} and e_{CD} are said to be mutually interfered. Without loss of generality, we assume that nodes A and C are sender nodes and B and D are receiver nodes.

Definition: e_{AB} and e_{CD} are said to be interfered with each other in G, if one of the following conditions holds.

1) same link: A = C and B = D.

2) same sender: A = C and B ≠ D.

3) same receiver: A ≠ C and B = D.

4) successive sender and receiver: (B = C and A ≠ D) or (A = D and B ≠ C).

5) hidden node: (A ≠ C and B ≠ D) and ($\exists$ e_{AD} or $\exists$ e_{CB}).

Case examples are shown in Fig. 3. The former four cases are the sources of primary interference and the last one is the source of secondary interference [3]. An additional interference called cumulative interference [19] may happen but is not considered in this research field. Cumulative interference is a kind of interference at a receiver node that consists of the cumulative power received from all the other sender nodes. For example, if there is no any single sender node that causes primary or secondary interference to a receiver node, but

there are many such sender nodes around, the cumulative noise level caused by these sender nodes still disturbs this receiver node. The cumulative interference can only be detected at physical layer, and hence we do not consider this issue in our discussion.

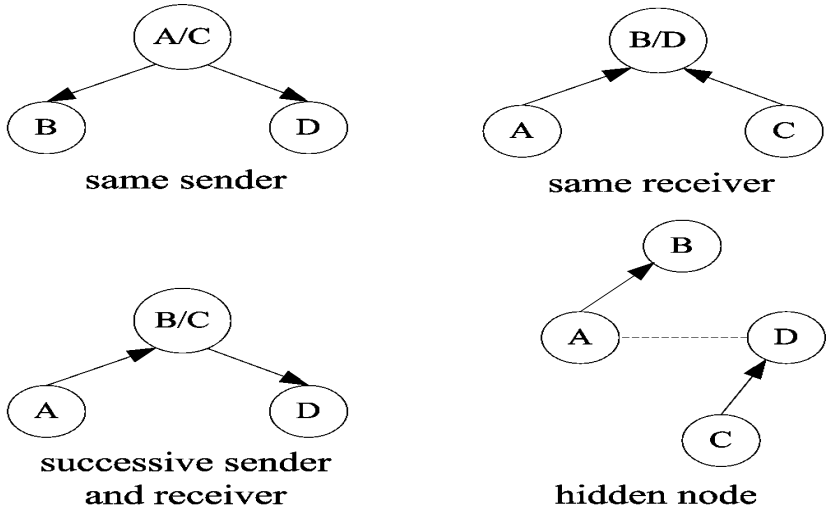


Fig. 3. Cases of interfered links.

Denote the cardinality of set $\bigcup_{v_i \in V} S_i$ by Δ. The centralized scheduling problem aims to find a valid minislot allocation with the minimum Δ value by maximizing the degree of minislot reuse. This problem can be divided into two parts: routing and scheduling. The routing problem is to derive G_{uR} and G_{dR} from G such that the subsequent schedule can work optimally. Given a routing tree, the scheduling problem is to allocate minislots such that user requirements are met and the total required minislots are minimized. This optimization problem is proven to be NP-complete in [5], and hence we resort to efficient heuristics.

3.2. Routing Tree Algorithm

The construction of G_{uR} and G_{dR} is based on the same algorithm. We first translate G into a layer graph $G_{\mathrm{L}} = (V, E)$. Each node v_i in G_{L} is associated with a layer number $l(v_i)$ which is equal to the minimal hop count of this node to v_0 (tree root). In G_{L}, we place nodes with the same layer number on the same level. The following notations explain node relationships in G_{L}.

- up_nodes(v_i) = {v_j | $v_j \in V$, $l(v_j) = l(v_i)-1$ and v_j is connected with v_i in G_{L}}

- down_nodes(v_i) = {v_j | $v_j \in V$, $l(v_j) = l(v_i)+1$ and v_j is connected with v_i in G_L}

The layer graph G_L preserves the shortest route paths to the tree root. The next job is to remove some edges from G_L to form G_{uR} or G_{dR}. The following notations explain node relationships in G_{uR} or G_{dR}.

- parent(v_i): parent node of v_i.
- children(v_i): set of child nodes of v_i.
- subtree_nodes(v_i): set of all nodes including v_i in the subtree rooted at v_i.
- level_nodes(i) = { v_j | $v_j \in V$, $l(v_j) = i$}

A routing tree is constructed by selecting a parent node for each node in G_L. The construction sequence is based on the node sequence in descending order of their bandwidth requirements. We aim to minimize the maximal node degree of the routing tree by examining down_degree values and distribute traffic from child nodes to parent nodes by examining sum_child_reqs values. The down_degree and sum_child_reqs values are defined as follows:

down_degree(v) = |down_nodes(v)|,

sum_child_reqs(v) =

$$\sum_{\forall x \in children(v)} N_{PDU}(g_{ui}) \cdot L_{PDU}(g_{ui}) \text{ or}$$

$$\sum_{\forall x \in children(v)} N_{PDU}(g_{di}) \cdot L_{PDU}(g_{di}) \quad .$$

The proposed tree construction is shown in Algorithm 1. The constructed tree is called a *traffic balanced tree* in the paper.

Algorithm 1: TreeConstruct(G_L)

Begin

$V' = V - \{v_0\}$.

$\forall$ v_i in V', set parent(v_i) = NULL.

$\forall$ v_i in V', set children(v_i) = ϕ.

Sort the nodes in V' in descending lexicographical order using the bandwidth requirement g_{ui} or g_{di} as the primary key and the node identifier (ID) as the secondary key.

Let v_i denote the ith node in V'.

For $i = 1 \sim |V'|$,

For each node v in up_nodes(v_i), compute the sum_child_reqs(v) and down_degree(v) values.

Use the following rules to break the tie in selecting a node x in up_nodes(v_i) to be the parent node of v_i, and then perform AddParentChild(x, v_i).

1) Select node x with the minimal sum_child_reqs value.

2) Select node x with the minimal down_degree value.

3) Select node x with the largest link data rate.

4) Select node x with randomness.

End for.

End.

Procedure: AddParentChild(x, v_i)

Begin

parent(v_i) = x.

children(x) = children(x) $\cup$ { v_i}.

End.

The complexity of this algorithm is $O(|V|\log|V| + |E|)$ which includes the cost to sorting $|V|$ nodes and examining all edges in E for leaving parent-child edges and removing the remaining unnecessary edges.

Fig. 4 shows an example construction of an uplink traffic balanced tree, where final tree links are represented by dashed lines. For simplicity, we have denoted the granted bandwidth to each node in the number of PDUs. Here, we assume that all PDUs have the same size of 100 bytes. The node sequence to be examined is 6, 4, 1, 8, 5, 3, 2, and 7 according to our sorting policy which gives high precedence to nodes with large bandwidth requirements and long distances away from the root node. Nodes v_6, v_4, and v_1 select the only node in their up_nodes sets as their parent nodes which are nodes v_3, v_1, and v_0 respectively. Node v_8 has two nodes v_5 and v_6 in its up_nodes set. Since there are no any nodes now to be child nodes

of these two nodes, their sum_child_reqs values are both zero (i.e., sum_child_reqs(v_5) = sum_child_reqs(v_6) = 0). Then, we compute their down_degree values and get down_degree(v_5) = 2 and down_degree(v_6) = 1. Therefore, v_8 selects v_6 as its parent node for balancing node degrees in the final tree. Next, v_5 needs to select one node from v_1 and v_2 as its parent node. v_1 already has a child node which is v_4, but v_2 has no such child nodes. We have sum_child_reqs(v_1) = 2×100 and sum_child_reqs(v_2) = 0, and hence v_2 is selected as the parent node. If v_1 is selected, uplink traffic from v_4 and v_5 is aggregated to v_1 and this reduces the chance to apply the minislot reuse rule. Nodes v_3 and v_2 select the only node in their up_nodes sets as their parent nodes which are the same node of v_0. Finally, node v_7 selects v_4 as its parent node due to down_degree(v_4) = 1 and down_degree(v_5) = 2.

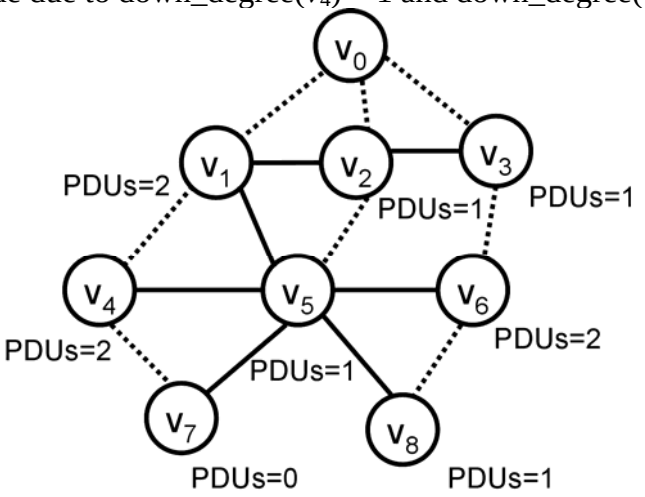


Fig. 4. Routing tree construction example.

3.3. Scheduling Algorithm

The scheduling problem becomes more complex if we consider the minislot allocation for uplink and downlink traffic together. Hence, we logically divide a data subframe into an uplink subframe and a downlink subframe as the case in PMP operation mode mentioned in the standard. Minislot allocations to G_{uR} and G_{dR} are separately performed within the uplink subframe and the downlink subframe. Here, we propose a scheduling algorithm that works with the same way to any uplink and downlink routing trees. However, the allocation output for a downlink routing tree needs an additional reverse operation.

Our proposed algorithm is modified from the line schedule proposed in [5]. Conceptually,

we select one path (called a line) from the root to a leaf node in the routing tree each time. Then the BGreedy algorithm [5] is applied to allocate one minislot to some links on this line. Meanwhile, we monitor from other tree links that are able to transfer data simultaneously using this same minislot. Then, another line is selected and the remaining processes are repeated. The BGreedy algorithm can produce an optimal allocation if the routing tree has a chain topology.

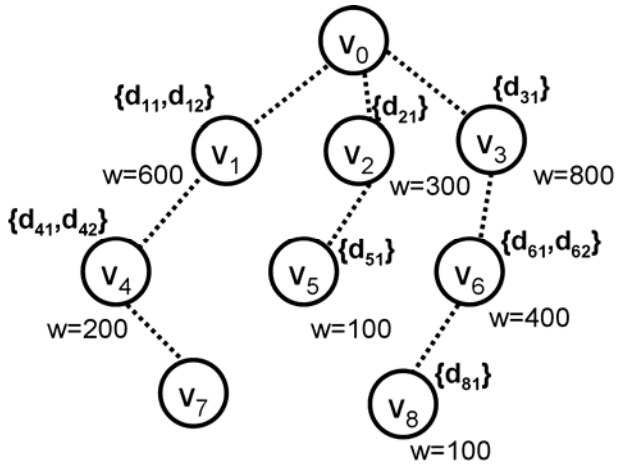


Fig. 5. An example traffic balanced tree.

The differences between our algorithm and the line schedule are as follows:

1) The line selection in the line schedule is done by iterating the path to each leaf node. For the example tree of Fig. 4, the lines to be examined are in the round sequence of paths from the root to leaf nodes v_7, v_5, and v_8. We consider in our algorithm the traffic load and forwarding overhead on each line. For example, the path from the root to leaf node v_8 is considered to be heavily loaded. In our scheme, this path is examined first and may be examined successively twice. Therefore, we propose a probability-based line selection method.

2) The node scanning to derive minislot reuse is based on the descending order of bandwidth requirements of nodes in the line schedule. For example, if edge e_{41} is allocated with a minislot in Fig. 4, three other edges e_{30}, e_{63}, and e_{86} can use this same minislot. e_{63} finally gets this minislot since v_6 has the highest bandwidth requirement among nodes v_3, v_6, and v_8. However, we consider the accumulated bandwidth requirement for each node from its descendent nodes in the tree; as a result, e_{30} gets the minislot due to having the highest accumulated requirement. This way can avoid upper-layer links to be bottleneck.

3) The line schedule considers only uplink traffic and assumes that all tree links have the same data rate. Our proposed algorithm is more general without these constraints.

In the following, we explain the scheduling steps by using an uplink routing tree shown in Fig. 5 which is derived from our traffic balanced tree in Fig. 4. The data rate of a thick dotted link is twice as large as that of a thin dotted link. We assume that all PDUs have the same size of 100 bytes. The *j*th PDU issued from v_i is represented by d_{ij}. The transmission of one PDU over a thick (thin) dotted link takes one (two) minislot(s).

At each minislot allocation, all *schedulable* nodes in the tree are searched and examined. The whole schedule is finished until all PDUs reach the BS.

Definition: Two nodes v_i and v_j in a routing tree are called mutually schedulable if they can transmit data to their individual parent nodes without causing any interference.

Definition: A node v is called schedulable to a set of nodes C, if v is mutually schedulable to each node in C.

Each tree node has buffer space to store PDUs that are waited for sending or forwarding. Let the total data size in the current buffer of v_i be $B(v_i)$ in bytes. Initially, $B(v_i)$ is equal to $N_{PDU}(g_{ui}) \cdot L_{PDU}(g_{ui})$. For example, $B(v_3) = 100$ and $B(v_6) = 200$. During a scheduling period, $B(v_i)$ is updated each time when one PDU is sent out to the parent node or received from one child node. Additionally, we compute the subtree weight of each tree node as follows:

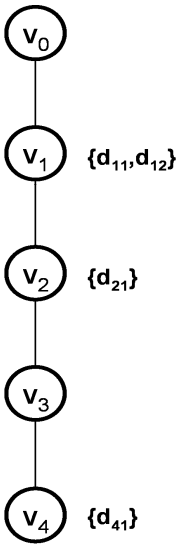
$$w(v_i) = \sum_{x \in subtree\_nodes(v_i)} B(x) \times (l(x) - l(v_i) + 1).$$

The subtree weight, which is the sum of products of the buffer size and the forwarding length, indicates the forwarding overhead of descendent nodes to the parent node. For example, $w(v_3)$ is 100×1 + 200×2 + 100×3 = 800. The subtree weights of the other nodes are indicated aside in the figure.

The procedure ProbLineSchedule illustrated in Algorithm 2 gives the detailed scheduling process. At each minislot allocation, the procedure SelectLine in Algorithm 3 is called to pick up one line in the routing tree. The line starts from the root node and is extended by repeatedly adding child nodes and ended at one leaf node. A child node having a larger subtree weight against its sibling nodes will be added with higher probability. The nodes along with this line excluding the root node are finally stored into set L.

For example, we start selecting a child node from the root node v_0. The probabilities of nodes v_1, v_2, and v_3 to be selected are 6/17, 3/17, and 8/17, respectively. Suppose here that node v_3 is selected. Next, we successively add two more child nodes v_6 and v_8 with probability of 1. Finally, the selected line is from v_0 to v_8 and L is { v_3, v_6, v_8}.

Next, we use the procedure BGreedy in Algorithm 4 to determine those schedulable nodes in L and store them into set C. BGreedy works like a bootstrapping process that always pulls one available PDU from the top of the line upward and concurrently pulls other not interfered PDUs underlying one by one. Due to the hidden node constraint, two links in a line that can transmit PDUs simultaneously should be separated with a distance of two hops. For example, the allocation for a chain topology of Fig. 6 takes seven minislots and only minislot one can be reused.



minislot	Schedule : (PDU,link)	
1	(d_{11},e_{10})	(d_{41},e_{43})
2	(d_{12},e_{10})	
3	(d_{21},e_{21})	
4	(d_{21},e_{10})	
5	(d_{41},e_{32})	
6	(d_{41},e_{21})	
7	(d_{41},e_{10})	

Fig. 6. Minislot allocation by BGreedy.

Finally, we look for the nodes in $V–L$ that are schedulable to C by using the procedure MaxSubtree in Algorithm 5. We search nodes layer by layer from the top downwards. At each layer, those nodes with non-empty buffers are examined in descending order of their subtree weights.

Algorithm 2: ProbLineSchedule(G_{uR})

Begin

SlotUse = the first minislot in an uplink subframe.

While (true),

$C = \phi$.

Perform SelectLine(L).

Construct G' by removing all nodes in L and related links from G_{uR}.

C = BGreedy(L).

C = MaxSubtree(G').

If C is empty, end the scheduling process.

For each node v in C,

Schedule the minislots starting from SlotUse for transmitting one PDU in the buffer of v to its parent node according to the burst profile between them.

Update the buffers of v and the parent node of v.

End for

SlotUse = the last minislot used in the above schedule + 1.

End while.

End.

Algorithm 3: SelectLine(L)

Begin

$L = \phi$.

next_node = v_0.

While (children(next_node) $\neq \phi$),

Compute $w(v)$ for each node v in children(next_node).
Compute $p(v)$ for each node v in children(next_node) by $p(v) = \frac{w(v)}{\sum_{\forall v \in children(next\_node)} w(v)}$.

Each node v has probability $p(v)$ to be selected, and here assume that v_k is selected.

next_node = v_k.

$L = L \cup \{v_k\}$.

End while.

End.

Algorithm 4: BGreedy(*L*)

Begin

Sort the nodes of L in ascending order of their layer numbers.

For each v_i from the first in L,

Put v_i into C, if $B(v_i) > 0$ and v_i is schedulable to set C.

End for.

Return set C.

End.

Algorithm 5: MaxSubtree(*G′*)

Begin

For each layer i ($i > 0$) from the top downwards,

Sort the nodes of level_nodes(i) in descending order of their subtree weights.

Let v_i be the ith node in level_nodes(i),

For each v_i from the first,

Put v_i into C, if $B(v_i) > 0$ and v_i is schedulable to set C.

End for.

End for.

Return set C.

End.

The final minislot allocation for the tree of Fig. 5 is shown in Table 1. At minislot one, the selected line is (v_0, v_8) and $L = \{v_3, v_6, v_8\}$. According to BGreedy, v_3 is added into set C. Then we run MaxSubtree to examine nodes v_1 and v_2 at layer 1, v_4 and v_5 at layer 2, and v_7 at layer 3. v_4 and v_7 are all schedulable nodes to v_3, but only v_4 is added into set C due to a non-zero subtree weight. Now, the result shows $C = \{v_3, v_4\}$, which indicates that links e_{30}

and e_{41} can activate together without interference. Therefore, we allocate two PDU transmissions of (d_{31}, e_{30}) and (d_{41}, e_{41}) at minislot one. It is updated the following subtree weights: $w(v_1) = 500$, $w(v_3) = 700$, and $w(v_4) = 100$.

minislot	Schedule : (PDU,link)		Line: (start , end)
1	(d_{31},e_{30})	(d_{41},e_{41})	(v_0, v_8)
2	(d_{61},e_{63})	(d_{11},e_{10})	(v_0, v_8)
3	(d_{12},e_{10})	(d_{62},e_{63})	(v_0, v_7)
4	(d_{61},e_{30})	(d_{42},e_{41})	(v_0, v_8)
5	(d_{21},e_{20})	(d_{81},e_{86})	(v_0, v_5)
6	(d_{41},e_{10})	(d_{81},e_{63})	(v_0, v_7)
7	(d_{51},e_{52})		(v_0, v_5)
8	(d_{62},e_{30})		(v_0, v_8)
9	(d_{51},e_{30})		(v_0, v_5)
10	(d_{42},e_{10})		(v_0, v_7)
11	(d_{81},e_{30})		(v_0, v_8)

Table 1. Complete minislot allocation.

At minislot two, the line (v_0, v_8) is selected again. BGreedy gets the allocation of (d_{61}, e_{63}) and MaxSubtree gets the allocation of (d_{11}, e_{10}). At minislot seven, the PDU transmission of (d_{51}, e_{52}) takes two successive minislots due to a slow-rate link. We continue this scheduling until there are no schedulable nodes.

For downlink traffic schedule, we simply use g_{di} to substitute g_{ui} in our proposed algorithms. Then the constructed routing tree becomes a downlink traffic balanced tree. The final minislot allocation order must be reversed additionally. In other works, the allocation to the last minislot becomes the allocation to the first one. The down allocation is placed within the downlink subframe. Certainly, a more condensed allocation can be derived by merging uplink and downlink minislot allocations, but this process takes time to complete.

The complexity of our scheduling algorithm is analyzed below. SelectLine takes cost $O(|V|)$ to compute subtree weights and produce a line path. BGreedy examines at most $|V|$ nodes to derive mutual schedulable nodes. Each examining takes a constant cost in checking

the hop distance between two nodes. MaxSubtree first takes totally cost $O(|V|\log|V|$ in sorting nodes layer by layer and then examines at most $|V|$ nodes for their schedulable features. Suppose that we have prepared a lookup table listing all non-interference edges with each edge in a routing tree. This table takes $|E|^2$, where $|E| = |V|-1$, pairwise comparisons to construct and cost $O(|E|)$ to look up. The final step in allocating minislots and updating the buffer takes at most cost $O(|V|)$. Therefore, the total complexity of our algorithm in allocating one minislot is $O(|V|\log|V| + |V|^2 + |V|)$.

3.4. QoS Schedule

To further support differentiate services on data transmission, we extend our scheduling algorithm by concerning about three QoS issues: service class, admission control, and fairness. Four service classes are defined in the IEEE 802.16 standard: Unsolicited Grant Service (UGS), real-time Polling Service (rtPS), non-real-time Polling Service (nrtPS), and Best Effort (BE). Each SS can initiate several traffic flows of different service classes together. These traffic flows may have various bandwidth requirements. Typically, we use the maximal sustained rate (r_i^{max}) and the minimum reserved rate (r_i^{min}) to specify the peak data rate and the minimum guaranteed data rate respectively. These four service classes have the following traffic characteristics:

UGS: $r_i^{max} = r_i^{min}$.

rtPS: $r_i^{max} >> r_i^{min}$ and a packet delay bound is specified.

rtPS: $r_i^{max} >> r_i^{min}$.

BE: $r_i^{min} = 0$ and r_i^{max} is not specified.

In the standard, the provisioning of these four service classes in mesh mode is not mentioned. Here we take advantage of some scheduling mechanisms design for PMP operations in our pervious work [20]. Initially, each SS aggregates the r_i^{min} (or r_{ui}^{min} and r_{di}^{min} with specified flow directions) values of all its traffic flows. Then, the SS sends this aggregated bandwidth requirement to the BS. After each schedule period, each SS measures the current queue size by counting the number of pending PDUs that are not served. Accordingly, an SS can temporarily increase its bandwidth requirement to deal with any traffic burst situations.

The granted bandwidth from the BS is shared by all traffic flows within an SS. A typical allocation technique is to adopt a multi-queue structure with separated inter-class and intra-class schedules as shown in Fig. 7. The inter-class schedule allocates the total granted bandwidth to each traffic class. Then, the intra-class schedule allocates the bandwidth to all traffic flows within each traffic class. Both these schedules can use a weighted fair function to distribute the available bandwidth. Moreover, an early-deadline-first policy is applied to rtPS traffic flows. The reader is referred to our work in [20] for more details.

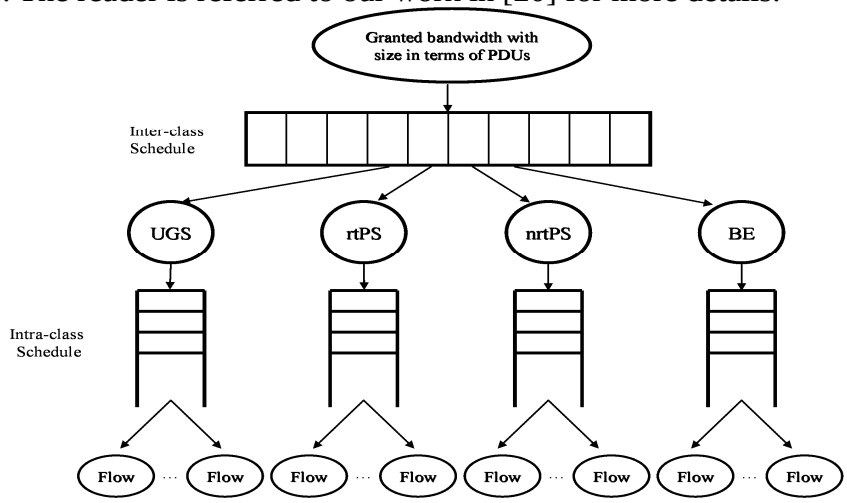


Fig. 7. Schedule architecture.

In order to meet minimum requirements of all traffic flows, an admission control mechanism is required for the BS not to grant too much bandwidth that exceeds the available one. The following condition should hold always:

$$upper\_demand = \sum_{\forall v_i \in V} \Big(\sum_{\forall e_{ij} \in P_{ui}} N_{PDU}(r_{ui}^{\min}) \cdot m(L_{PDU}(r_{ui}^{\min}), b_{ij}) + \sum_{\forall e_{ij} \in P_{di}} N_{PDU}(r_{di}^{\min}) \cdot m(L_{PDU}(r_{di}^{\min}), b_{ij}) \Big) \ ,$$

$$upper\_demand \le 256 \times \text{MSH-CSCH-DATA-FRACTION}$$

where upper_demand calculates the maximal number of required minislots by considering no minislot reuse.

The final issue is concerning about fairness among SSs. According to the standard, each SS should reduce its bandwidth requirement evenly when total minislot requirements exceed the upper bound associated with the centralized schedule. This becomes unfair if the requirements of SSs are highly skewed. To solve this, one possible way is to announce a

shaping factor to each SS by the BS. Those SSs that have been granted with larger accumulative bandwidths get larger shaping values and shrink more bandwidth requirements than the other SSs when the network is saturated.

4. PERFORMANCE EVALUATION

We evaluate the performance of various routing tree construction and minislot allocation algorithms through simulation programs written in C++. Most of existing solutions provide only uplink traffic schedule with the constraint that all communication links in a routing tree have the same data rate. For a reasonable comparison, we use the same scenario in our performance evaluation.

4.1. Simulation Model

We consider a network environment with area of 1,200 m×1,200 m. All SSs with number of 30 ~ 70 (default 50) are randomly placed into this area. The communication range of either an SS or a BS is varied from 150 m to 350 m (default 150 m). We generate uplink bandwidth requirements in number of PDUs for SSs using a normal distribution with a mean of 5 and a standard deviation from 1 to 20 (default 20). All PDUs are of the same size and the transmission of one PDU takes one minislot. We assume that the BS fully grants all bandwidth requirements and the number of minislots in a data subframe is an unbounded value. In other words, we can allocate more than 256 minislots.

The routing tree construction algorithms examined in our experiments include interference-aware tree, maximum parallelism tree, min max degree tree, and our proposed traffic balanced tree. The minislot allocation algorithms that are examined include interference-aware schedule, nearest schedule, minimum interference schedule, greedy schedule, line schedule, and our proposed probability-based line schedule. All experimental results are the average values of 10 runs. Two cost metrics are used: 1) the number of minislots required in the final allocation (the Δ value), and 2) the channel utilization ratio. The latter metric introduced in [9] is defined as (3). A higher channel utilization ratio indicates a more compact minislot allocation.

Channel utilization ratio = $\sum_{v_i \in V} (N_{PDU}(g_{ui}) \times l(v_i)) / (|V| \times \Delta)$ (3)

4.2. Simulation Results

In the comparison of routing tree construction algorithms, the same probability-based line schedule is used. In this experiment, we first vary the number of SSs in the network and the result is shown in Fig. 8. The performance order is traffic balanced > maximum parallelism > min max degree > interference-aware. The interfere-aware tree performs worse due to inappropriate measurement of interference degrees using the number of neighboring nodes. Both min max degree and maximum parallelism trees are derived from a layer graph. The min max degree tree cannot evenly distribute aggregated traffic to different parent nodes, because the traffic load on each link is not considered. The maximum parallelism tree is built in top-down and layer-by-layer method. However, the traffic balanced tree is built according to a link sequence from heavy traffic load to light traffic load. Moreover, aggregated traffic is more evenly distributed to different parent nodes in this tree. The traffic balanced tree gets 4.8%, 5.7%, and 33.1% improvements against the other three algorithms in performance order on average.

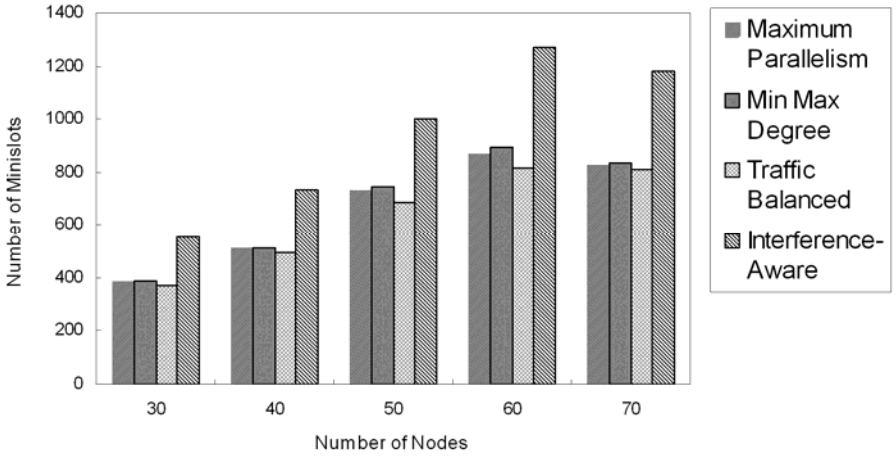


Fig. 8. Routing comparison under different number of nodes.

We observe the change of standard derivation values on bandwidth requirements in Fig. 9. The traffic balanced tree gets the most improvement as the deviation value is 1 and the improvement values are 10.2%, 10.2%, and 43.5% against the other three algorithms. When the standard deviation is increased and even larger than the mean value, the bandwidth re-

quirements of most SSs are zero. In such a sparse network, the improvement values become 3.6%, 4.1%, and 31.0%.

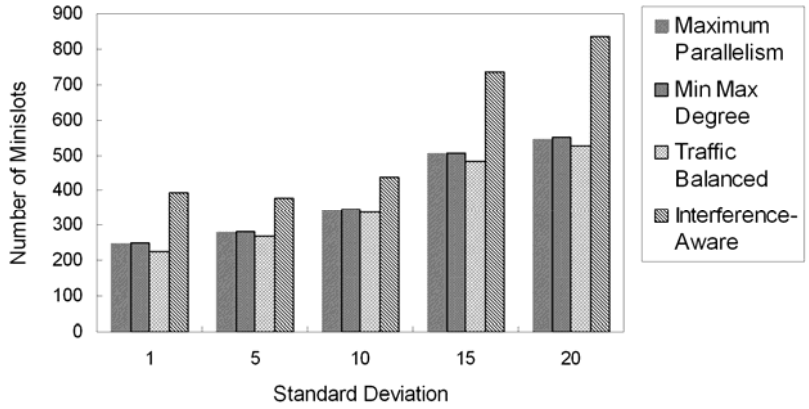


Fig. 9. Routing comparison under different deviation values.

Finally, we change the communication range and the result is shown in Fig. 10. The traffic balanced tree gets 6.3%, 7.1%, and 43.3% improvements. As can be seen, the improvement values slightly become small when the communication range is greater than 300 m. This is because the edge density becomes large and most links are mutually interfered. The degree of minislot reuse is therefore small.

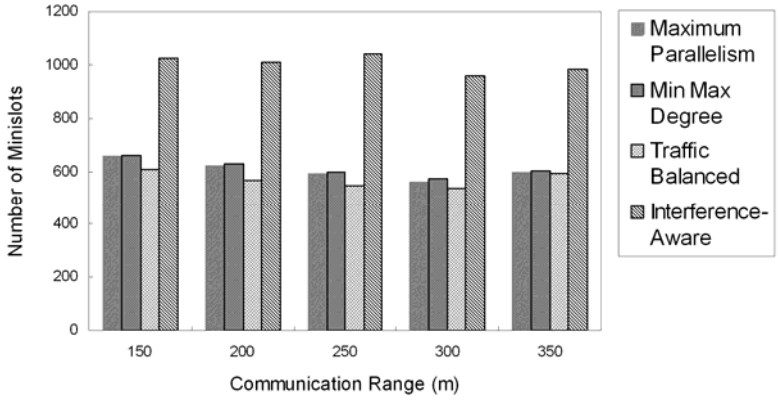


Fig. 10. Routing comparison under different communication ranges.

In the comparison of different allocation algorithms, we use the same traffic balanced tree. We first change the number of SSs and the result is shown in Fig. 11. The performance order is probability-based > line schedule > (interference-aware, nearest, greedy) > minimum interference. Our proposed algorithm gets 1.8% and 18.6% improvements against line

schedule and minimum interference schedule on average. The minimum interference schedule tends to allocate minislots to the links first that are far-away from the BS and hence is a bottom-up scheduling scheme. The other algorithms all belong to top-down scheduling schemes. Therefore, a top-down scheme is more efficient than a bottom-up one.

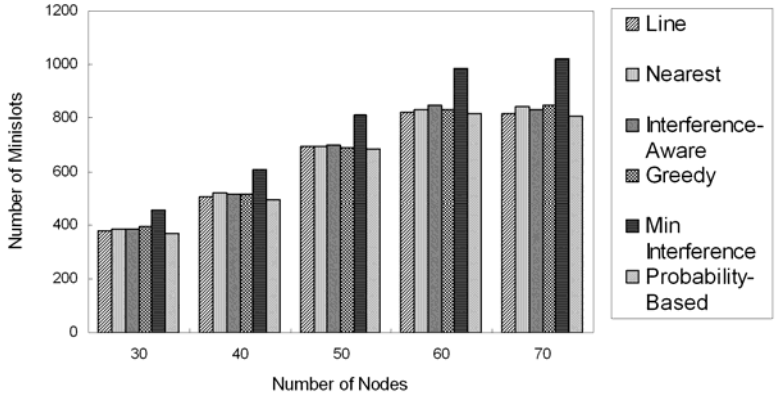


Fig. 11. Scheduling comparison under different number of nodes.

Next, we observe the change of standard deviations in Fig. 12. The performance order is the same as the previous one. Our proposed algorithm gets 1.8% and 20.2% improvements against line schedule and minimum interference schedule. The influence of communication ranges on the performance is shown in Fig. 13. The improvement values are 1.4% and 18.3% against the second best one and the worst one. Again, these improvement values are decreased as the communication range is increased.

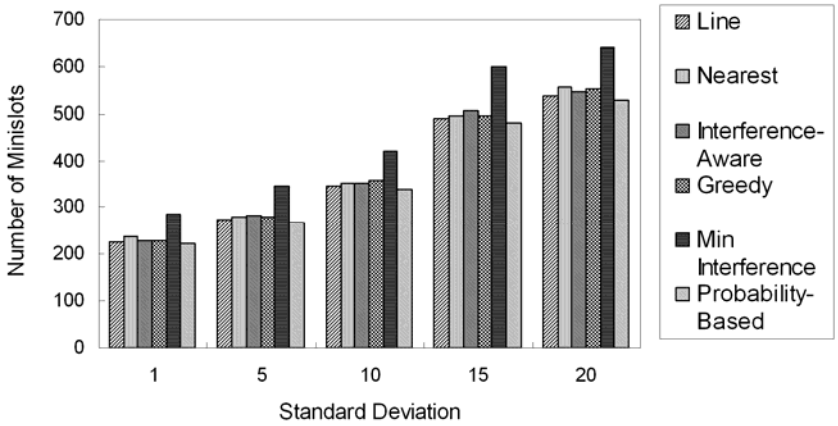


Fig. 12. Scheduling comparison under different deviation values.

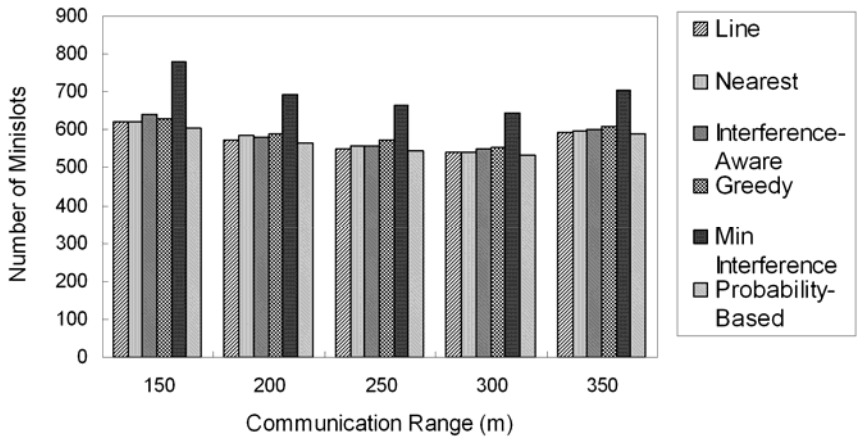


Fig. 13. Scheduling comparison under different communication ranges.

From the above observations, our proposed routing tree construction algorithm brings more improvement than the proposed scheduling algorithm. The core technique of our scheduling algorithm is similar to the line schedule which has performed close to the optimal solution, so the improvement is not significant. Moreover, we do not specially generate any experimental samples in which some route paths are very heavily loaded, and hence the benefit of our proposed line selection method is not highlighted.

Finally, we compare our proposed algorithms with other two combinations of existing algorithms in terms of channel utilization. A higher channel utilization ratio indicates a higher degree of minislot reuse. The experimental results are shown in Figs. 14, 15 and 16. Our proposed algorithm suit gets 8%, 9%, and 7% improvements against the other two algorithm suits on average under different SS numbers, standard deviations, and communication ranges respectively.

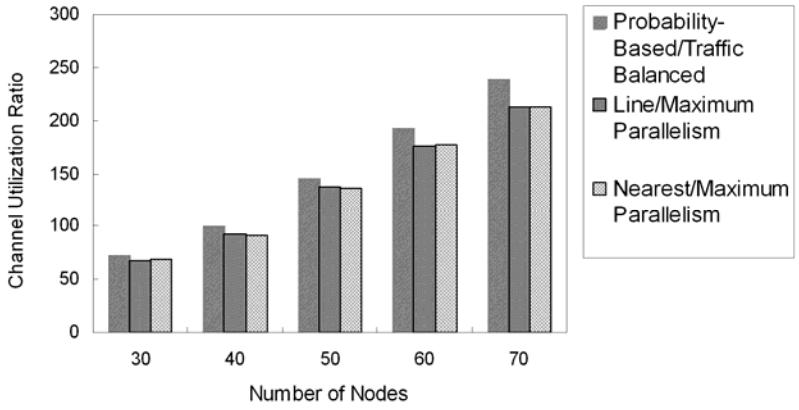


Fig. 14. Channel utilization under different number of nodes.

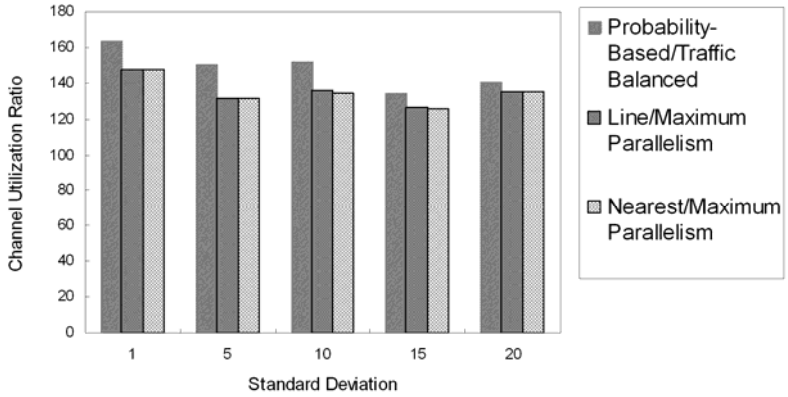


Fig. 15. Channel utilization under different deviation values.

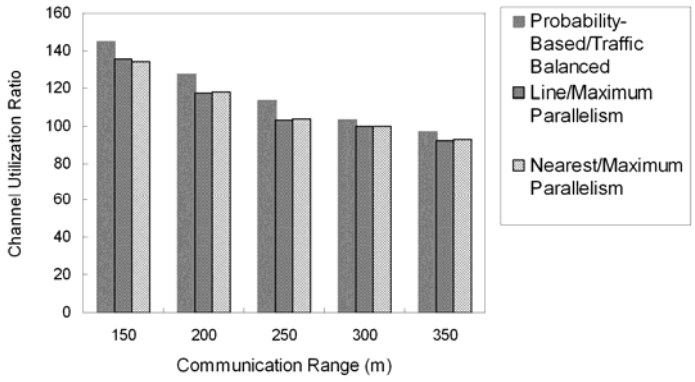


Fig. 16. Channel utilization under different communication ranges.

In Tables 2 and 3, the time complexities of these algorithms are summarized. The time

complexity of our proposed scheduling algorithm is similar to the line schedule but the proposed tree construction algorithm is lower than the maximum parallelism one. Therefore, our proposed algorithms incur less overhead for providing better performance.

Algorithm	Complexity
Inference-Aware	$O(\|V\| + \|E\|)$
Maximum Parallelism	$O(\|E\|^2)$
Min Max Degree	$O(\|E\|)$
Traffic Balanced	$O(\|V\|\log\|V\| + \|E\|)$

Table 2. Time complexities of routing algorithms.

Algorithm	Complexity
Inference-Aware	$O(\|V\|^2 + \|V\|)$
Nearest	$O(\|V\|^2 + \|V\|)$
Min Interference	$O(\|V\|^2 + \|V\|)$
Greedy	$O(\|V\|^2)$
Line	$O(\|V\|\log\|V\| + \|V\|^2)$
Probability-Based	$O(\|V\|\log\|V\| + \|V\|^2 + \|V\|)$

Table 3. Time complexities of scheduling algorithms.

5. CONCLUSIONS

This paper addresses a channel minislot allocation problem in the IEEE 802.16 mesh network. We focus on the centralized scheduling mechanism which has good features on resource guarantee and contention-free data transmission and is suitable for Internet traffic flows. Though the scheduling problem is NP complete, we provide efficient routing and scheduling algorithms to this problem. The proposed algorithms consider both load balancing and fairness issues. Experiment results show that these algorithms can achieve higher channel utilization than existing ones. More importantly, our algorithms are practical and flexible due to the support of different link data rates and both uplink and downlink traffic flows. Moreover, we provide some basic methods to deal with QoS issues such as service class, admission control, and fairness. We would extend our work in the future to a network

environment with multiple radios and channels.

REFERENCES

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

[2] C. Eklund, R. B. Marks, S. Ponnuswamy, K. L. Stanwood, and N. J.M. van Waes, "WirelessMAN: Inside the IEEE 802.16 Standard for Wireless Metropolitan Area Networks," *Standards Information Network*, IEEE Press, 2006.

[3] N. A. Abu Ali, A. E. M. Taha, H. S. Hassanein, and H. T. Mouftah, "IEEE 802.16 Mesh Schedulers: Issues and Design Challenges," *IEEE Network*, vol. 22, no. 1, pp. 58-65, 2008.

[4] H. Y. Wei, S. Ganguly, and R. Izmailov, "Interference-Aware IEEE 802.16 WiMax Mesh Networks," *in Proc. IEEE Vehicular Technology Conf. (VTC)*, pp.3102-3106, 2005.

[5] F. Jin, A. Arora, J. Hwang, and H. A. Choi, "Routing and Packet Scheduling for Throughput Maximization in IEEE 802.16 Mesh Networks," *in Proc. IEEE Intl. Conf. on Broadband Networks*, Sept. 2007.

[6] L. W. Chen, Y. C. Tseng, D. W. Wang, and J. J. Wu, "Exploiting Spectral Reuse in Resource Allocation, Scheduling, and Routing for IEEE 802.16 Mesh Networks," *in Proc. IEEE Vehicular Technology Conf. (VTC)*, pp. 1608-1612, Sept. 2007.

[7] H. Shetiya and V. Sharma, "Algorithm for Routing and Centralized Scheduling in IEEE 802.16 Mesh Networks," *in Proc. IEEE Wireless Communications and Networking Conf. (WCNC)*, pp. 147-152, Apr. 2006.

[8] H. Shetiya, and V. Sharma, "Algorithm for Routing and Centralized Scheduling to Provide QoS in IEEE 802.16 Mesh Networks," *in Proc. ACM Intl. Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP)*, Oct. 2005.

[9] B. Han, W. Jia, and L. Lin, "Performance Evaluation of Scheduling in IEEE 802.16 Based Wireless Mesh Networks," *in Proc. IEEE Intl. Conf. on Mobile Adhoc and Sensor Systems (MASS)*, pp. 789-794, Oct. 2006.

[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. Intl. Wireless Communications and Mobile Computing Conf. (IWCMC)*, Jul. 2006.

[11] P. Djukic and S. Valaee, “Link Scheduling for Minimum Delay in Spatial Re-use TDMA,” *in Proc. IEEE INFOCOM Conf.*, pp. 28-36, May 2007.

[12] P. Djukic and S. Valaee, “Scheduling Algorithms for 802.16 Mesh Networks,” *Wi-Max/MobileFi: Advanced Research and Technology*, pp. 267-288, Auerbach Publications, 2007.

[13] A. Raniwala, K. Gopalan, and T. C. Chiueh, “Centralized Channel Assignment and Routing Algorithms for Multi-Channel Wireless Mesh Networks,” *Mobile Computing and Communication Review*, vol. 8, no. 2, pp. 50-65, Apr. 2004.

[14] P. Kyasanur, J. So, C. Chereddi, and N. H. Vaidya, “Multichannel Mesh Networks: Challenges and Protocols,” *IEEE Wireless Communications*, vol. 13, no.2, pp. 30-36, Apr. 2006.

[15] Y. Chen, S. Liu, and C. Chen, “Channel Assignment and Routing for Multi-Channel Wireless Mesh Networks Using Simulated Annealing,” *in Proc*. *IEEE GLOBECOM Conf.*, pp. 1-5, Nov. 2006.

[16] S. Avallone and I. Akyildiz, “A Channel Assignment Algorithm for Multi-Radio Wireless Mesh Networks,” *in Proc*. *IEEE ICCCN Conf*, pp. 1034-1039, Aug. 2007.

[17] A. Raniwala and T. Chiueh, “Architecture and Algorithms for an IEEE 802.11-based Multi-Channel Wireless Mesh Network,” *in Proc. IEEE INFOCOM Conf.*, pp. 2223-2234, March 2005.

[18] C. Lin and C. Chou, “Route-Aware Load-Balanced Resource Allocation for Wireless Mesh Networks,” *in Proc*. *IEEE WCNC Conf.*, pp. 3093-3098, Mar. 2007.

[19] P. Gupta and P. R. Kumar, “The Capacity of Wireless Networks,” *IEEE Trans. Info. Theory*, vol. 46, no. 2, pp. 388-404, 2000.

[20] S. C. Lo and Y. Y. Hong, “A Novel QoS Scheduling Approach for IEEE 802.16 BWA Systems,” *in Proc. Intl. Conf. on Communication Technologies* (ICCT), November 2008.

Shou-Chih Lo received the B.S. degree in computer science from National Chiao Tung University, Taiwan, in 1993, and the Ph.D. degree in computer science from National Tsing Hua University, Taiwan, in 2000. He joined the Computer & Communication Research Center at National Tsing Hua University in 2000 as a Postdoctoral Fellow. Since 2004, he has been with the Department of Computer Science and Information Engineering of the National Dong Hwa University in Taiwan, where he is currently an Associate Professor. His current research interests are in the area of mobile and wireless networks with emphasis on mobility management, MAC protocols, and broadcast services designs.

Lyu-Chen Ou received the B.S. degree in information and computer engineering from Chung Yuan Christian University, Taiwan, in 2005, and the M.S. degree in computer science and information engineering from National Dong Hwa University, Taiwan, in 2008.