



## II. RELATED WORK

Transmission scheduling in WiMAX-based multihop networks with omni-directional antennas has been studied recently. Different centralized heuristic algorithms have been proposed for scheduling and/or routing in [3], [13], [23], [24] with the objective of maximizing spatial reuse and enhancing fairness. In [6], a distributed algorithm was presented to provide fair end-to-end bandwidth allocation for single-radio, multi-channel WiMAX mesh networks. In [12], the authors introduced a low-complexity distributed scheduling-based MAC protocol that can support all feasible arrival rates in a wireless backhaul network. In [22], Sundaresan *et al.* showed that the scheduling problem to exploit diversity gains alone in a 2-hop 802.16j-based mesh network is NP-hard, and provided polynomial-time approximation algorithms to solve it. They also proposed a heuristic algorithm to exploit both spatial reuse and diversity. In [7], the authors studied a similar problem, and proposed an easy-to-compute upper bound on the optimum and three fast heuristic algorithms.

Smart antennas have also received tremendous research attention. MAC protocols have been proposed in [5], [14] for 802.11-based ad-hoc networks with switched beam antennas. In [21], Sundaresan *et al.* presented a constant factor approximation algorithm for DOF assignment and a distributed algorithm for joint DOF assignment and scheduling in ad-hoc networks with DAA antennas. The authors of [19] presented a centralized algorithm as well as a distributed protocol for stream control and medium access in ad-hoc networks with MIMO links. A constant factor approximation algorithm was proposed for a similar problem in [16]. A unified representation of the physical layer capabilities of different types of smart antennas, and unified medium access algorithms were presented in [20]. In [9], Hu and Zhang devised a MIMO-based MAC protocol. They also studied its impact on routing and characterized the optimal hop distance that minimizes end-to-end delay. Cross-layer optimization for MIMO-based wireless networks has also been studied in [2], [15]. In [2], Bhatia and Li presented a centralized algorithm to solve the joint routing, scheduling and stream control problem subject to fairness constraints.

The difference between our work and these related works are summarized as follows: 1) As mentioned before, due to the interference suppression feature of DAA antennas, the scheduling problem here is significantly different from the omni-directional antenna based scheduling problems. 2) Our objective is to improve end-to-end throughput and fairness. However, the scheduling problems studied in [5], [14], [16], [19], [20], [21] aim at maximizing single-hop throughput or minimizing the frame length. 3) The routing tree and transmission schedule computed by our algorithms have certain provably good properties which cannot be supported by the heuristic algorithms reported in [6], [13], [23], [24].

## III. SYSTEM MODEL

We focus on DAA antennas and use an antenna operation model similar as that in [21]. In order to enable communications between a transmitter and receiver pair (link)  $(v_i, v_j)$ , a DOF needs to be assigned for communications at each

end. Except for the DOF assigned for communications, the remaining  $(K - 1)$  DOFs can be used by the transmitter to cancel its interference to other nodes by forming nulls at corresponding directions. Similarly, the receiver can employ its remaining  $(K - 1)$  DOFs to suppress interference from other nodes.

The WiMAX standard [1] adopts a scheduling-based MAC protocol, in which the time domain is divided into minislots, and multiple minislots are grouped together to form a frame. Each frame is composed of a control subframe and a data subframe. The control subframe is used to exchange control messages. Data transmissions occur in the data subframe, which includes  $T$  minislots with fixed durations, and is further partitioned into an uplink subframe and a downlink subframe with  $T^u$  and  $T^d$  minislots respectively. Note that  $T^u$  does not have to be equal to  $T^d$ . In our simulations, they were set to be proportional to the uplink and downlink bandwidth demands. Other methods can also be used to determine  $T^u$  and  $T^d$ . The WiMAX MAC protocol [1] supports both centralized scheduling and distributed scheduling. Centralized scheduling is the focus of this paper.

We consider a static wireless backhaul network with a BS and  $(n - 1)$  SSs. All transmissions are conducted on a single common channel. These nodes will form a spanning tree rooted at the BS for routing. Each node has a single transceiver and a DAA antenna with  $K$  DOFs. We model the network using a *communication graph*  $G(V, E)$ , in which each node corresponds to the BS or an SS. If all nodes are placed on a plane with no blockage in between and transmit at the same fixed power level, then each node will have a uniform transmission range  $R_T$ , which is usually much larger than the transmission range supported by omni-directional antennas. In this case,  $(v_i, v_j) \in E$  if  $\|v_i - v_j\| \leq R_T$ . Of course, we can determine whether there exists a link between a pair of nodes with consideration for other factors such as Line Of Sight (LOS) and terrain effects. In addition, for each node  $v_i$ , we can identify a set  $N_i$  of neighboring nodes potentially interfering with  $v_i$ , using the method introduced in [21]. Briefly,  $v_j \in N_i$  if the Signal to Interference and Noise Ratio (SINR) at receiver  $v_j$  (or  $v_i$ ) will drop below the threshold due to  $v_i$  (or  $v_j$ ) unless nullified. We may also use a more conservative but simpler method, i.e.,  $v_j \in N_i$  if  $\|v_i - v_j\| \leq R_I$ , where  $R_I$  is the interference range, normally 2 - 3 times  $R_T$ . If two links  $e = (v_i, v_j)$  and  $e' = (v_{i'}, v_{j'})$  are incident on a common node, we say there exists *primary interference* between them. In this case, there is no way that they can be active concurrently due to the half-duplex (a transceiver can only transmit or receive at one time), unicast (a transmission only involves a single intended receiver) and collision-free (two transmissions intended for a common receiver cannot happen at the same time) constraints. If nodes  $v_i, v_j, v_{i'}$  and  $v_{j'}$  are distinct but  $v_j \in N_{i'}$  or  $v_{j'} \in N_i$ , there exists *secondary interference* between them. A DOF can be assigned at either the transmitter or the receiver to suppress secondary interference.

## IV. PROBLEM DEFINITION

Our scheme is to construct a spanning tree rooted at the BS for routing and schedule transmissions along the tree according to the traffic demands. The tree is constructed beforehand and

will be used for a relatively long time period. This scheme is compatible with the WiMAX MAC protocol [1] and is suitable for a wireless backhaul network in which the topology usually does not change but the traffic demands change over time. A joint routing and scheduling scheme may require a significant modification to the current WiMAX standard which only includes MAC and PHY protocols, and may lead to a larger overhead in the case where traffic demands change frequently. We summarize the primary notations in Table I.

TABLE I  
NOTATIONS

$\mathbf{A}^u/\mathbf{A}^d$	The aggregated uplink/downlink bandwidth allocation vector
$\mathbf{B}^u/\mathbf{B}^d$	The uplink/downlink bandwidth allocation vector
$G(V, E)$	The communication graph
$H[v]$	The layer of node $v$ in the routing tree
$I^p(v)/I^s(v)$	The primary/secondary interference value of node $v$
$I^s(e)$	The secondary interference value of link $e$
$I^b(h)$	The secondary interference bound of layer $h$
$K$	The number of DOFs at each node
$n/m$	The number of nodes/links in $G$
$N_i$	The set of nodes which can potentially interfere with $v_i$
$p_i$	The index of the parent node of $v_i$
$\mathbf{Q}^u/\mathbf{Q}^d$	The uplink/downlink bandwidth demand vector
$R_T/R_I$	The transmission/interference range
$T/T^u/T^d$	The number of minislots in a frame/ uplink subframe/downlink subframe
$\mathbf{S}^u/\mathbf{S}^d$	The uplink/downlink satisfaction ratio vector
$V_h/E_h$	The set of nodes/links in layer $h$
$\Gamma$	The scheduling matrix
$\Lambda$	The DOF assignment matrix

It is well-known that interference has a significant impact on network performance [11]. So we will construct a low-interference tree, which hopefully can provide good throughput for any bandwidth demands. Note that both uplink and downlink traffic use the same tree for routing. So every link in the communication graph  $G$  is treated as an *undirected* link for the tree construction. If an SS can directly connect to the BS or another SS, we prefer not to use other SSs as relay because additional hops will introduce longer delay, and more importantly, result in more severe interference. We can set the transmission range  $R_T$  to be less than the actual range to avoid having long links when constructing  $G$ . Therefore, once  $G$  is given, we can easily determine on which layer of the routing tree an SS  $v_i$  should appear, by conducting a Breadth First Search (BFS) on  $G$ .  $V_h$  and  $E_h$  denote the set of nodes in layer  $h$  and the set of links between layer  $h$  and  $(h-1)$  respectively. Moreover,  $h_{\max}$  denotes the total number of layers (the BS  $v_0$  is not considered as a layer), i.e., the height of the tree. The tree construction problem is essentially the problem of determining which node in layer  $(h-1)$  should serve as the parent node for each node  $v_i$  in layer  $h$ . For all the SSs in the first layer, their parent must be the BS ( $v_0$ ).

We need to differentiate primary and secondary interference because primary interference can only be resolved by scheduling but secondary interference may also be eliminated by carefully assigning DOFs. Given a tree, the primary interference value of a node  $v_i$ ,  $I^p(v_i)$ , is defined as the total number of links incident to  $v_i$  on the tree. We define the secondary interference value of  $v_i$  as  $I^s(v_i) = |N_i| - 1$ ,

and the secondary interference value of link  $e = (v_i, v_j)$  as  $I^s(e) = \max\{I^s(v_i), I^s(v_j)\}$ . In addition, we define a secondary interference bound for each layer  $h > 1$ ,  $I^b(h) = \max\{I^s(e_b), K-1\}$ , where  $e_b$  is the bottleneck link, i.e., if all links with secondary interference values greater than or equal to  $I^s(e_b)$  in  $G$  are removed, at least one node in  $V_h$  will be disconnected from nodes in  $V_{h-1}$ .

**Definition 1 (ITCP):** The **Interference aware Tree Construction Problem (ITCP)** seeks a spanning tree  $Y$  rooted at the BS, such that in each layer  $h > 1$ ,  $I_{\max}^p = \max_{v \in V_h} I^p(v)$  is minimized subject to the constraint that  $I^s(e) \leq I^b(h)$ ,  $\forall e \in Y \cap E_h$ .

By solving the ITCP, a balanced routing tree will be constructed. Moreover, potential secondary interference on this tree is likely to be eliminated by properly assigning DOFs.

The scheduling problems considered here involve both transmission scheduling and DOF assignment. The input includes a spanning tree with the BS  $v_0$  as the root,  $(n-1)$  SSs  $\{v_1, \dots, v_{n-1}\}$ , their bandwidth demands  $\mathbf{Q}^u = [q_1^u, \dots, q_{n-1}^u]$  and  $\mathbf{Q}^d = [q_1^d, \dots, q_{n-1}^d]$  for uplink and downlink respectively, and the uplink/downlink subframe sizes  $T^u/T^d$ .  $q_i^u$  denotes the number of minislots SS  $v_i$  needs to transmit its uplink traffic. Note that if  $v_i$  is not a leaf node,  $q_i^u$  includes the bandwidth needed for itself but does not include the bandwidth requested by any of its descendants on the tree.

We define an uplink scheduling matrix  $\Gamma$  and a corresponding DOF assignment matrix  $\Lambda$ .  $\Gamma_{i,j}^t = 1$  if link  $(v_i, v_j)$  is active in minislot  $t$ ;  $\Gamma_{i,j}^t = 0$  otherwise. Note that only  $(n-1)$  links on the given tree will be considered for scheduling.  $\Lambda_{i,j}^t = 1$  if  $v_i$  assigns a DOF to point at  $v_j$  for interference suppression in minislot  $t$ ;  $\Lambda_{i,j}^t = 0$  otherwise.  $\Lambda$  has nothing to do with DOFs assigned for communications since  $\Gamma_{i,j}^t = 1$  implies that one DOF at  $v_i$  and one DOF at  $v_j$  need to be assigned for communications. A scheduling matrix  $\Gamma$  and a DOF assignment matrix  $\Lambda$  are said to be *feasible* if 1) there does not exist primary or secondary interference in every minislot; and 2)  $\sum_{j=0}^{n-1} \Lambda_{i,j}^t \leq K-1$ ,  $0 \leq i \leq n-1$ ,  $1 \leq t \leq T^u$ . We also define an *uplink bandwidth allocation vector*  $\mathbf{B}^u = [b_1^u, \dots, b_{n-1}^u]$ , its corresponding *satisfaction ratio vector*  $\mathbf{S}^u = [s_1^u, \dots, s_{n-1}^u] = [b_1^u/q_1^u, \dots, b_{n-1}^u/q_{n-1}^u]$ , and its corresponding *aggregated bandwidth allocation vector*  $\mathbf{A}^u = [a_1^u, \dots, a_{n-1}^u]$ , where  $b_i^u$  indicates the actual bandwidth (the number of minislots in each frame) allocated to  $v_i$  for uplink traffic generated at  $v_i$ , and  $a_i^u$  indicates the actual bandwidth allocated to  $v_i$  for uplink traffic generated at  $v_i$  and all of its descendants. A bandwidth allocation vector  $\mathbf{B}^u$  is said to be *feasible* if there exists a feasible scheduling matrix  $\Gamma$ , such that  $\sum_{t=1}^{T^u} \Gamma_{i,p_i}^t \geq a_i^u$ ,  $1 \leq i \leq n-1$ .

**Definition 2 (USP):** The **Uplink Scheduling Problem (USP)** seeks a feasible uplink bandwidth allocation vector  $\mathbf{B}^u$  and its corresponding satisfaction ratio vector  $\mathbf{S}^u$ , along with a corresponding feasible scheduling matrix  $\Gamma$  and DOF assignment matrix  $\Lambda$  such that the minimum satisfaction ratio,  $\min_{1 \leq i \leq n-1} s_i^u$  is maximized.

In the USP, we try to maximize the minimum satisfaction ratio for the purpose of fairness. Similarly, we can define the Downlink Scheduling Problem (DSP). Due to space limitations and redundancy, we omit the details.

## V. THE TREE CONSTRUCTION ALGORITHM

We present an optimal algorithm (Algorithm 1) to solve the ITCP. Its input includes the communication graph  $G$  ( $G$  is treated as an undirected graph for routing), an array  $H$  which gives the layer of each node, and the number of layers  $h_{\max}$ . An array  $Parent$  is the output, which specifies the parent node for each SS.

---

### Algorithm 1 Solve-ITCP( $G, H, h_{\max}$ )

---

```

Step 1 forall  $h \in \{1, \dots, h_{\max}\}$   $V_h \leftarrow \{v | H[v] = h\}$ ;
      endforall
       $h \leftarrow h_{\max}$ ;
      forall  $v \in V_1$   $Parent[v] \leftarrow v_0$ ; endforall
      forall  $v \in V \setminus V_1$   $Parent[v] \leftarrow \text{null}$ ; endforall
Step 2 if ( $h = 1$ ) return  $Parent$ ;
Step 3  $d_{\max} \leftarrow \max_{v \in V_{h-1}} d_v$ , where  $d_v$  is the number of
       $v$ 's neighbors in  $V_h$ ;
       $lb \leftarrow 0$ ;  $ub \leftarrow d_{\max}$ ;
Step 4 while ( $lb \leq ub$ )
       $mid \leftarrow \lfloor \frac{lb+ub}{2} \rfloor$ ;
      Construct the auxiliary graph  $G'(V', E')$ ;
      Apply the Ford-Fulkerson algorithm [4] on  $G'$  to
      find the maximum flow  $f_{\max}$  from  $s$  to  $d$ 
      and the corresponding link flow allocation  $Flow$ ;
      if ( $f_{\max} = |V_h|$ )  $ub \leftarrow mid - 1$ ;
      else  $lb \leftarrow mid + 1$ ; endif
      endwhile
Step 5 forall  $e = (u, v) \in E'$ 
      if ( $Flow[e] = 1$  and  $u \neq s$  and  $v \neq d$ )
           $Parent[u] \leftarrow v$ ; endif
      endforall
Step 6  $h \leftarrow h - 1$ ;
      goto Step 2;

```

---

Our algorithm constructs the tree in a bottom-up fashion. It starts from nodes on layer  $h_{\max}$ , and selects a node from nodes in layer  $(h - 1)$  as the parent node for each node in layer  $h$  in Steps 3–5. In Step 4, nodes in layer  $(h - 1)$  are connected to some nodes in layer  $h$ , while the maximum primary interference value of nodes in layer  $h$  is minimized using a binary search. The auxiliary graph  $G'(V', E')$  in Step 4 is constructed as follows:  $V' = V_h \cup V_{h-1} \cup \{s, d\}$ , where  $s$  and  $d$  are virtual source and sink nodes respectively. For each  $u \in V_h$ , we create a directed link with a capacity of 1 from  $s$  to  $u$ . For each  $u \in V_h$  and  $v \in V_{h-1}$ , we create a directed link  $e$  with a capacity of 1 from  $u$  to  $v$ , if  $(u, v) \in E$  and  $I^s(e) \leq I^b(h)$ . Finally, for each  $v \in V_{h-1}$ , we create a directed link with a capacity of  $mid$  from  $v$  to  $d$ . In Step 5, we compute the parent assignment for nodes in layer  $h$  according to the link flow allocation  $Flow$ . We use a simple example to illustrate the construction of the auxiliary graph in Fig. 2. In the figure, the secondary interference values of links  $(C, A)$ ,  $(D, A)$ ,  $(E, A)$ ,  $(D, B)$ ,  $(E, B)$  and  $(F, B)$  are assumed to be no greater than the corresponding bound  $I^b(h)$ .

*Theorem 1:* Algorithm 1 optimally solves the ITCP in  $O(mnh_{\max} \log \delta_{\max})$  time, where  $n$ ,  $m$ ,  $h_{\max}$  and  $\delta_{\max}$  are the number of nodes, the number of links, the number of layers and the maximum node degree of  $G$  respectively.

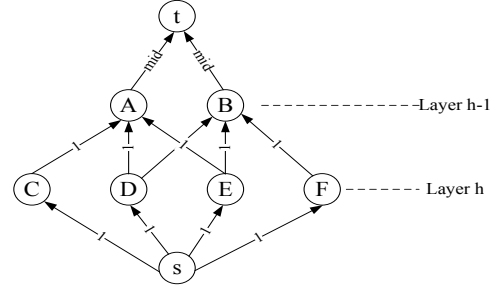


Fig. 2. The auxiliary graph  $G'$

*Proof:* As mentioned before, the ITCP is essentially the problem of determining which node should be selected as the parent node for each node  $v$  in the next layer. Since the secondary interference value of each link crossing two layers can be pre-determined, the constraint of the ITCP can be satisfied by including only links with secondary interference values less than or equal to  $I^b(h)$  in the auxiliary graph. Therefore, in each layer  $h$ , the problem reduces to determine a parent node assignment in the bipartite graph given by  $V_{h-1}$ ,  $V_h$  and the links in between, such that each node in  $V_h$  is connected to exactly one node in  $V_{h-1}$  and the maximum primary interference value of nodes in  $V_{h-1}$  is minimized, which is achieved by the binary search in Step 4. Note that the primary interference value of a non-leaf node is actually equal to the number of its children plus one. In each iteration of the binary search in Step 4, we need to check if there exists an assignment such that each node in  $V_h$  is connected to exactly one node in  $V_{h-1}$  and the children of each node in  $V_{h-1}$  are no more than  $mid$ .

Next, we show that there exists such an assignment if and only if the maximum  $s$ - $d$  flow is equal to  $|V_h|$ . First, it is well-known that an augmenting path based maximum flow algorithm such as the Ford-Fulkerson algorithm [4] can always find a maximum flow whose corresponding link flows are all integers if the capacity of each link is an integer. If the maximum flow found by the Ford-Fulkerson algorithm is  $|V_h|$ , then there must be exactly one unit flow going into each node in  $V_h$  from  $s$  since the capacity of every link between  $s$  and a node  $u \in V_h$  is 1. According to the flow conservation constraint and the integer flow claim mentioned above, there must be exactly one unit flow going from every node  $V_h$  to a node in  $V_{h-1}$ , which actually leads to a feasible parent node assignment. Moreover, the capacities of the links connecting nodes in  $V_{h-1}$  to  $d$  are set to  $mid$ , which ensures that each node in  $V_{h-1}$  has no more than  $mid$  children.

In Algorithm 1, Step 1 takes  $O(n)$  time for initialization. The time complexity of Step 3 depends on the number of nodes and links in the two consecutive layers, which are obviously bounded by  $m$  and  $n$ . So Step 3 takes  $O(m + n)$  time. The Ford-Fulkerson algorithm can find the maximum flow within  $O(|E'|f_{\max})$  time, where  $f_{\max} \leq |V_h|$  for our problem. Moreover,  $d_{\max} \leq (\delta_{\max} - 1)$ . So Step 4 can be done within  $O(\log(\delta_{\max} - 1)|V_h||E'|) = O(mn \log \delta_{\max})$  time. It is easy to see Step 5 takes  $O(|E'|) = O(m + n)$  time. Steps 3–5 will be executed  $(h_{\max} - 1)$  times. The total time complexity of Algorithm 1 is therefore  $O(mnh_{\max} \log \delta_{\max})$ . ■

Our algorithm is time-efficient in practice because the

number of links between two consecutive layers is usually much less than  $m$ , and  $h_{\max}$  and  $\delta_{\max}$  are normally small.

## VI. THE SCHEDULING ALGORITHMS

In this section, we present algorithms to solve the scheduling problems. Since the uplink and downlink traffic are scheduled for transmissions independently in different subframes according to the WiMAX MAC protocol [1], *we only discuss the USP and the corresponding algorithms in the following*. The downlink scheduling simply follows.

The link scheduling problems in a multihop wireless network (even only with omni-directional antennas) are usually NP-hard [17]. Therefore, in the first part of this section, we consider a *special case* of the USP, where each node has a relatively large number of DOFs but a relatively small number of potential interferers in its neighborhood, such that there exists a trivial DOF assignment which can eliminate all potential secondary interference. For example, if the number of DOFs in each node  $v_i$ ,  $K \geq \lceil \frac{N_{\max}}{2} \rceil + 1$ , where  $N_{\max} = \max_{0 \leq i \leq n-1} |N_i|$ , then there exists a trivial secondary interference free DOF assignment since half of total secondary interference can be taken care of by DOFs at the active receivers and another half can be dealt with by DOFs at the active transmitters. Therefore, in this special case, only the primary interference needs to be addressed for transmission scheduling. In the second part, we propose a heuristic algorithm for the general case where both primary and secondary interference need to be addressed.

### A. The Scheduling Algorithm for the Special Case

The basic idea of the proposed algorithm is to identify the *bottleneck node* in each step, compute the corresponding bandwidth allocation for both the bottleneck node and its descendants based on their demands, and then remove them from the tree (by setting their demands to 0 in our algorithm). This procedure is repeated until all the nodes are removed. The algorithm for solving the special case USP is formally presented as Algorithm 2, whose input includes the bandwidth demand vector  $\mathbf{Q} = [q_1, \dots, q_{n-1}]$ , the number of minislots available for uplink traffic  $T^u$  and the routing tree  $Y$ .

---

#### Algorithm 2 Solve-Special-USP( $\mathbf{Q}, T^u, Y$ )

---

```

Step 1  $P \leftarrow \{i | v_i \text{ is a non-leaf node on } Y\};$ 
      forall  $i \in P$   $T_i \leftarrow T^u$ ; endforall
Step 2  $\gamma_0 \leftarrow \text{Schedule-BS}(\mathbf{Q}, T_0);$ 
      forall  $i \in P \setminus \{0\}$ 
         $\gamma_i \leftarrow \text{Schedule-SS}(\mathbf{Q}, T_i, i);$ 
      endforall
Step 3  $j \leftarrow \text{argmin}_{i \in P} \gamma_i;$ 
       $D \leftarrow \{i | v_i \text{ is a descendant of } v_j \text{ on } Y\};$ 
       $C \leftarrow \{i | v_i \text{ is an ancestor of } v_j \text{ on } Y\};$ 
       $B_j \leftarrow \sum_{k \in D} b_k + b_j;$ 
      forall  $i \in C$   $T_i \leftarrow T_i - B_j$ ; endforall
      forall  $i \in D \cup \{j\}$   $q_i = 0$ ; endforall
      if  $(\mathbf{Q} \neq \mathbf{0})$  goto Step 2; endif

```

---

Algorithm 2 starts with the BS and check the SSs one by one to find the bottleneck node using Algorithms 3 and 4 in

Step 2. The details are discussed later. In Step 3, the bottleneck node and all of its descendants are removed from the tree, and the number of free minislots in each of its ancestors is updated. The procedure is repeated until all nodes are scheduled.

---

#### Algorithm 3 Schedule-BS( $\mathbf{Q}, T_0$ )

---

```

Step 1  $\mathbf{B} \leftarrow \mathbf{0};$   $q_{\text{total}} \leftarrow \sum_{1 \leq i \leq n-1} q_i;$ 
      if  $(q_{\text{total}} \leq T_0)$ 
         $\mathbf{B} \leftarrow \mathbf{Q}$ ; return 1;
      endif
Step 2  $\gamma \leftarrow \frac{T_0}{q_{\text{total}}};$ 
      forall  $k \in \{1, \dots, n-1\}$ 
         $b_k \leftarrow \lfloor \gamma q_k \rfloor;$   $s_k \leftarrow \frac{b_k}{q_k};$  ( $s_k \leftarrow 1$  if  $q_k = 0$ )
      endforall
       $T_{\text{rem}} \leftarrow T_0 - \sum_{1 \leq k \leq n-1} b_k;$ 
Step 3  $j \leftarrow \text{argmin}_{1 \leq k \leq n-1} s_k;$ 
      if  $(T_{\text{rem}} = 0)$  return  $s_j$ ; endif
       $b_j \leftarrow b_j + 1;$   $s_j \leftarrow \frac{b_j}{q_j};$   $T_{\text{rem}} \leftarrow T_{\text{rem}} - 1;$ 
      goto Step 3;

```

---



---

#### Algorithm 4 Schedule-SS( $\mathbf{Q}, T_i, i$ )

---

```

Step 1  $\mathbf{B} \leftarrow \mathbf{0};$   $D \leftarrow \{j | v_j \text{ is a descendant of } v_i \text{ on } Y\};$ 
       $D' \leftarrow D \cup \{i\};$ 
       $q_{\text{total}} \leftarrow q_i + 2 \sum_{k \in D} q_k;$ 
      if  $(q_{\text{total}} \leq T_i)$ 
        forall  $k \in D'$   $b_k \leftarrow q_k$ ; endforall
        return 1;
      endif
Step 2  $\gamma \leftarrow \frac{T_i}{q_{\text{total}}};$ 
      forall  $k \in D'$ 
         $b_k \leftarrow \lfloor \gamma q_k \rfloor;$   $s_k \leftarrow \frac{b_k}{q_k};$  ( $s_k \leftarrow 1$  if  $q_k = 0$ )
      endforall
       $T_{\text{rem}} \leftarrow T_i - b_i - 2 \sum_{k \in D} b_k;$ 
Step 3  $j \leftarrow \text{argmin}_{k \in D'} s_k;$ 
      if  $(T_{\text{rem}} = 0)$  or  $(T_{\text{rem}} = 1$  and  $b_i = q_i)$ 
        return  $s_j$ ;
      endif
      if  $(j = i$  or  $T_{\text{rem}} = 1)$ 
         $b_i \leftarrow b_i + 1;$   $s_i \leftarrow \frac{b_i}{q_i};$   $T_{\text{rem}} \leftarrow T_{\text{rem}} - 1;$ 
      else
         $b_j \leftarrow b_j + 1;$   $s_j \leftarrow \frac{b_j}{q_j};$   $T_{\text{rem}} \leftarrow T_{\text{rem}} - 2;$ 
      endif
      goto Step 3;

```

---

Algorithms 3 and 4 are similar, which not only test whether the BS or a particular SS is the bottleneck node, but also compute a corresponding bandwidth allocation. Note that once the bandwidth allocation is determined, it is trivial to find a corresponding transmission schedule and DOF assignment in the special case. In both algorithms,  $q_{\text{total}}$  gives the total number of minislots required for the traffic that needs to go through the corresponding node (including the traffic generated by itself and all its descendants). At every non-leaf node  $v_i$  (including the BS  $v_0$ ), if  $q_{\text{total}} \leq T_i$ , then both its demand and its descendants' demands can be fully satisfied. Otherwise,

the bandwidth is allocated to nodes in the subtree rooted at  $v_i$  according to the ratio  $\frac{q_{\text{total}}}{T^u}$ . After that, the remaining minislots (if there are any) are allocated to nodes in the ascending order of their current satisfaction ratios, until no more allocation can be made. After such an allocation procedure, the minimum satisfaction ratio of nodes on the subtree rooted at  $v_i$  can be obtained and returned, which we call *the effective satisfaction ratio of  $v_i$* . The non-leaf node with the minimum effective satisfaction ratio is identified as the *bottleneck node*. Note that the number of minislots that can be allocated for a node  $v_k$  at its different ancestors are different. Basically, the nodes in the upper layers are more likely to be the bottleneck node since they need to handle more relay traffic.

We present Algorithm 3 for the BS and Algorithm 4 for the SSs since the bandwidth allocation in the BS is different from that in an SS. A non-leaf SS  $v_i$  needs to allocate bandwidth for traffic generated by itself as well as relay traffic generated by its descendants. Therefore, in order to provide one minislot to one of its descendants  $v_k$ , two minislots need to be arranged for  $v_k$  at  $v_i$ , one for link  $(v_{h_i}, v_i)$  and another for link  $(v_i, v_{p_i})$ , where  $h_i$  and  $p_i$  are the indices of  $v_i$ 's child relaying  $v_k$ 's traffic and  $v_i$ 's parent node respectively. However, the bandwidth allocation in the BS is simpler since it does not have a parent node.

**Theorem 2:** Algorithm 2 computes a bandwidth allocation vector  $\mathbf{B}$  with max-min satisfaction ratio in  $O(n^3 \log n)$  time.

*Proof:* In Algorithm 2, if the BS is the bottleneck node, i.e.,  $j = 0$ , Algorithm 3 will be executed once to compute a bandwidth allocation vector. We show that Algorithm 3 always finds a bandwidth allocation vector  $\mathbf{B}$  with max-min satisfaction ratio. If all bandwidth demands can be 100% satisfied, Algorithm 3 terminates at Step 1 and can obviously find a bandwidth allocation vector  $\mathbf{B}$  with max-min satisfaction ratio. Otherwise, the algorithm terminates when the number of remaining free minislots  $T_{rem} = 0$ . In this case, if there exists another bandwidth allocation vector  $\mathbf{B}'$  with a larger minimum satisfaction ratio, i.e.,  $s'_{\min} > s_{\min}$ , then  $\exists k, b'_k \geq b_k + 1$ . Since there are no free minislots, increasing the bandwidth allocation of some node  $v_k$  must lead to decreasing the bandwidth allocation of another node  $v_j$ , i.e.,  $\exists j, b'_j \leq b_j - 1$ . Therefore, we have

$$s'_j = \frac{b'_j}{q_j} \leq \frac{b_j - 1}{q_j} = \frac{\lfloor \gamma q_j \rfloor - 1}{q_j} \leq \gamma - \frac{1}{q_j} \quad (1)$$

In Step 2 of Algorithm 3, since each  $b_k$  is rounded down to the nearest integer  $\lfloor \gamma q_k \rfloor$ ,  $\forall k, (\gamma - s_{\min})q_k \leq 1$ . Therefore,  $\gamma - \frac{1}{q_j} \leq s_{\min}$ . Combining this with (1), we have  $s'_j \leq s_{\min}$ . This contradicts the assumption that  $s'_{\min} > s_{\min}$ . Therefore, there does not exist a bandwidth allocation vector  $\mathbf{B}'$  with a larger minimum satisfaction ratio, i.e.,  $s'_{\min} > s_{\min}$ .

Next, we consider the case where the bottleneck node  $v_j$  is an SS, i.e.,  $j \neq 0$ . Similarly as above, we can prove that Algorithm 4 can compute a bandwidth allocation vector with max-min satisfaction ratio for the subtree rooted at  $v_j$  based on the minislot availability. The detailed proof is omitted due to space limitation. Let  $s_{\min}$  be the overall minimum satisfaction ratio found by Algorithm 2. Since  $v_j$  is the bottleneck node,  $s_{\min} = w_j$ , where  $w_j$  is the effective satisfaction ratio of  $v_j$ .

It is impossible to find another bandwidth allocation vector  $\mathbf{B}'$  with minimum satisfaction ratio  $s'_{\min} > s_{\min}$ . Otherwise, the effective satisfaction ratio of  $v_j$  in  $\mathbf{B}'$ :  $w'_j \geq s'_{\min} > s_{\min} = w_j$ . This contradicts our proof that  $w_j$  is maximized for the subtree rooted at  $v_j$ . Therefore, Algorithm 2 always finds a bandwidth allocation vector  $\mathbf{B}$  with max-min satisfaction ratio.

In Algorithm 3, both Step 1 and Step 2 take  $O(n)$  time. Step 3 takes  $O(n \log n)$  time to process  $s_k$  in order. Therefore, Algorithm 3 takes  $O(n \log n)$  time. Similarly, Algorithm 4 can be done in  $O(n \log n)$  time. In Algorithm 2, it takes  $O(n)$  time for initialization in Step 1. In Step 2, Algorithm 4 is executed  $O(n)$  times, each of which takes  $O(n \log n)$  time. Hence, the total running time of this step is  $O(n^2 \log n)$ . Step 3 takes  $O(n)$  time. Since Step 3 removes at least one node from the spanning tree  $Y$ , the loop composed of Steps 2 and 3 will be executed  $O(n)$  times. Therefore, the time complexity of Algorithm 2 is  $O(n^3 \log n)$ . ■

In most cases, the bottleneck node is either the BS or an SS in the first layer. If the SSs are labeled in a top-down fashion, Step 2 of Algorithm 2 will be executed only a few times. Therefore, the running time of Algorithm 2 is only  $O(n^2 \log n)$  in practice.

### B. The Scheduling Algorithm for the General Case

We present an efficient heuristic algorithm (Algorithm 5) to solve the USP in the general case. It includes a subroutine that can optimally determine whether a set of links can be active simultaneously, which is not trivial in the context of DAA antennas since DOFs can be allocated to suppress interference and enable concurrent transmissions.

---

#### Algorithm 5 Solve-USP( $\mathbf{Q}, T^u, L^s$ )

---

```

Step 1  $\Gamma \leftarrow \mathbf{0}; \Lambda \leftarrow \mathbf{0}; \mathbf{X} \leftarrow \mathbf{0}; t \leftarrow 1;$ 
      forall  $i \in \{1, \dots, n-1\}$ 
         $a_i \leftarrow \sum_{k \in D_i} q_k + q_i;$ 
      endforall
Step 2 Sort  $L^s$  in the ascending order of link satisfaction ratios;  $L \leftarrow \emptyset;$ 
Step 3 forall  $e = (v_i, v_j) \in L^s$ 
       $(A, flag) \leftarrow \text{AssignDOF}(L, e);$ 
      if  $(flag = \text{TRUE})$ 
         $L \leftarrow L \cup \{e\}; \Gamma_{i,j}^t \leftarrow 1; x_i \leftarrow x_i + 1;$ 
        if  $(\frac{x_i}{a_i} = 1)$   $L^s \leftarrow L^s - \{e\};$  endif
        if  $(A \neq \emptyset)$ 
          forall  $(k, l) \in A$   $\Lambda_{k,l}^t \leftarrow 1;$  endforall
          forall  $(k, l) \notin A$   $\Lambda_{k,l}^t \leftarrow 0;$  endforall
        endif
      endif
      endforall
Step 4  $t \leftarrow t + 1;$ 
      if  $(t \leq T^u)$  goto Step 2; endif

```

---

Algorithm 5 computes a scheduling matrix  $\Gamma$  with a scheduling period of  $T^u$  minislots, a DOF assignment  $\Lambda$  and a corresponding link bandwidth allocation vector  $\mathbf{X}$  ( $x_i$  indicates the bandwidth allocated to link  $(v_i, v_{p_i})$ ) for the given bandwidth demands  $\mathbf{Q}$  and a set  $L^s$  of links on the routing tree  $Y$ . The algorithm is a greedy algorithm that

tries to pack as many links as possible in a minislot in the ascending order of their satisfaction ratios. Note that the satisfaction ratio of a link is equal to the number of minislots that have been allocated to it divided by the total number of minislots needed for transmitting both local and relay traffic. The core part of this algorithm is the subroutine AssignDOF (Algorithm 6), which determines whether a set of links can be active concurrently and gives a feasible DOF assignment if the answer is YES.

---

**Algorithm 6** AssignDOF( $L, e$ )

---

```

Step 1 if ( $\exists l \in L, e$  is incident to  $l$ )
    return ( $\emptyset$ , FALSE);
endif
if ( $e = (v_i, v_j)$  does not have secondary interference
with any link in  $L$ )
    return ( $\emptyset$ , TRUE);
endif
Step 2 Construct the auxiliary graph  $G'(V', E')$ ;
Apply the Ford-Fulkerson algorithm on  $G'$  to find a
maximum flow  $f_{\max}$  from  $s$  to  $d$  and the correspond-
ing link flow allocation  $Flow$ ;
Step 3 if ( $f_{\max} < |TR|$ ) return ( $\emptyset$ , FALSE); endif
forall  $e' = (x, y) \in E_2$  (where  $x$  corresponds to
node  $v_k$ ,  $y$  corresponds to node pair  $(v_h, v_l)$ )
    if ( $Flow[e'] = 1$ )
        if ( $k = h$ )  $A \leftarrow A \cup \{(h, l)\}$ ;
        else  $A \leftarrow A \cup \{(l, h)\}$ ; endif
    endif
endforall
return ( $A$ , TRUE);

```

---

In Step 1, we check whether there exists primary interference between  $e$  and a link in  $L$ . If so, there is no way to activate them concurrently via DOF assignment. Moreover, if there is no secondary interference between  $e$  and any link in  $L$  (including the case where  $L = \emptyset$ ), we immediately know  $e$  can be active concurrently with  $L$ . Otherwise, the auxiliary directed graph  $G' = (V', E')$  is constructed to find a feasible DOF assignment, which consists of three types of vertices. In the following, we use  $T_r$  and  $R_v$  to denote the set of transmitters and receivers corresponding to link set  $L \cup \{e\}$  respectively. We also define a node pair set  $TR = \{(v_i, v_j) | v_i \in T_r, v_j \in R_v, v_i \in N_j, (v_i, v_j) \notin L \cup \{e\}\}$ . Each of the first type of vertices in  $V'$  corresponds to a node in  $T_r \cup R_v$ . Each of the second type of vertices in  $V'$  corresponds to a node pair in  $TR$ . The corresponding vertex sets are denoted as  $V_1$  and  $V_2$  respectively.  $V'$  also includes two virtual vertices  $s$  and  $d$ . There is an edge from  $s$  to each vertex in  $V_1$  with a capacity of  $(K - 1)$ , where  $K$  is the number of DOFs at each node. In  $G'$ , there are two edges going to each vertex in  $V_2$  corresponding to node pair  $(v_i, v_j)$ , one from the vertex corresponding to  $v_i$  and another from the vertex corresponding to  $v_j$ , both of which have a capacity of 1. There is also an edge from each vertex in  $V_2$  to  $d$  with a capacity of 1. The corresponding edge sets are denoted as  $E_1$ ,  $E_2$  and  $E_3$  respectively. Hence, we have  $V = V_1 \cup V_2 \cup \{s, d\}$  and  $E = E_1 \cup E_2 \cup E_3$ .

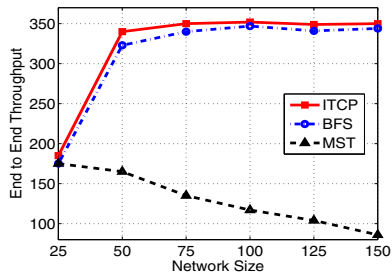
In  $G'$ , vertices in  $V_2$  actually correspond to potential secondary interference. We create two edges for such a vertex because potential secondary interference can be eliminated by assigning a DOF at either the corresponding transmitter or receiver. Every node has  $K$  DOFs and  $(K - 1)$  of them can be used to suppress secondary interference. That is why the capacity of each edge in  $E_1$  is set to  $(K - 1)$ . As mentioned before, the Ford-Fulkerson algorithm can always find a maximum flow whose corresponding link flows are all integers if the capacity of each link is an integer. Therefore, if the Ford-Fulkerson algorithm can find a maximum flow of  $|TR| (|V_2|)$  in  $G'$ , then there exists a feasible DOF assignment such that all potential secondary interference can be cancelled. Otherwise, link  $e$  cannot be active concurrently with links in  $L$ . Obviously, Algorithm 5 is a polynomial-time algorithm.

## VII. PERFORMANCE EVALUATION

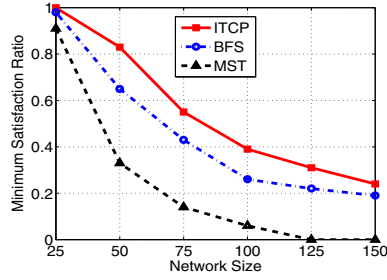
The performance of the proposed algorithms was evaluated via simulation. In the simulations, the nodes were uniformly deployed within a  $4 \times 8$  km<sup>2</sup> rectangular region, with a BS at the top-left corner. The number of nodes (network size) varied from 25 to 150, with 25 as the step size. According to [1], the number of minislots per frame was set to 1024. The uplink and downlink demands of an SS were set to a random number uniformly distributed in [5, 10] and [10, 20] respectively. The transmission and interference ranges were set to 1 km and 3 km respectively.

Since our work is the first to address WiMAX scheduling with smart antennas, we compared our scheduling algorithms with the first-fit algorithm and a trivial solution. The first-fit algorithm is a typical greedy algorithm, which tries to pack as many links with unsatisfied bandwidth demands as possible in the first minislot in a top-down fashion without violating the interference constraints, and then repeats this procedure for the next minislot until all minislots are used. The trivial solution is mentioned in the WiMAX standard [1], which does not allow spatial reuse (i.e., only one link is active in each minislot). In terms of routing, we compared the trees constructed by our algorithm with the MSTs and BFS trees. The end-to-end throughput, the minimum satisfaction ratio and the well-known Jain's fairness index [10]  $f(s_1^u, s_2^u, \dots, s_n^u) = \frac{(\sum_{i=1}^n s_i^u)^2}{n \sum_{i=1}^n (s_i^u)^2}$  were used as the performance metrics, where  $s_i^u$  is the uplink satisfaction ratio of node  $v_i$ .

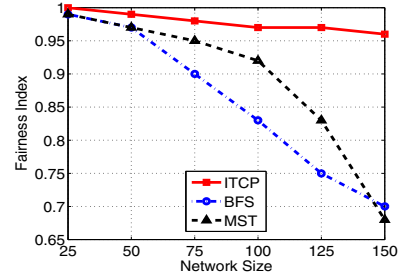
In the first scenario, we compared different tree construction algorithms and scheduled the transmissions using our scheduling algorithm proposed for the general case. The corresponding results are presented in Fig. 3. In scenarios 2 and 3, we evaluated the performance of different scheduling algorithms for the special and general cases respectively. Our algorithm for solving the ITCP was always used to construct the routing tree. The results are presented in Figs. 4 and 5 respectively. In scenario 4, we evaluated the performance of different complete solutions (scheduling + routing). Refer to Fig. 6 for the results. For scenarios 1, 3, and 4, the number of DOFs at each node was set to  $K = 3$ . Each number presented in the figures is the average over 100 simulation runs. In each run, a network is randomly generated. In these figures, "USP" stands for our uplink scheduling algorithm for the special and general cases



(a) End-to-end throughput

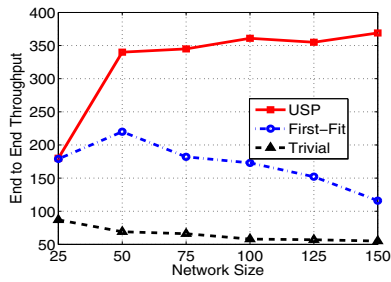


(b) Minimum satisfaction ratio

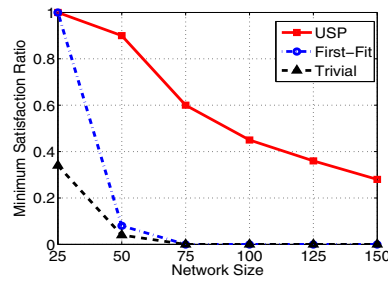


(c) Fairness index

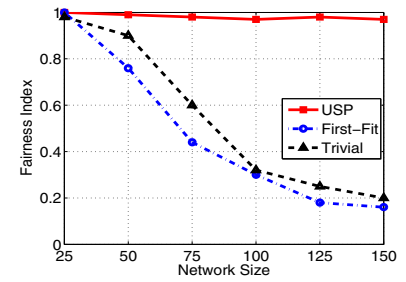
Fig. 3. The tree construction algorithms



(a) End-to-end throughput

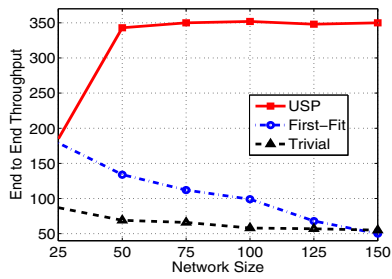


(b) Minimum satisfaction ratio

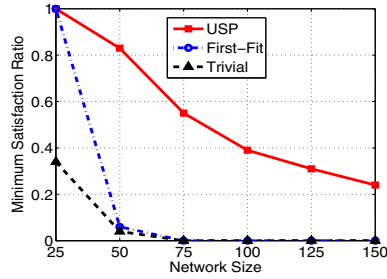


(c) Fairness index

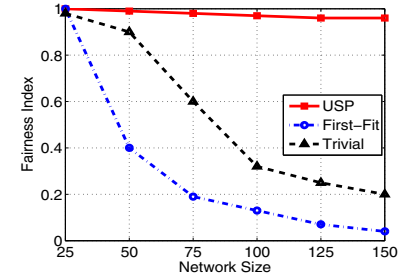
Fig. 4. The scheduling algorithms for the special case



(a) End-to-end throughput

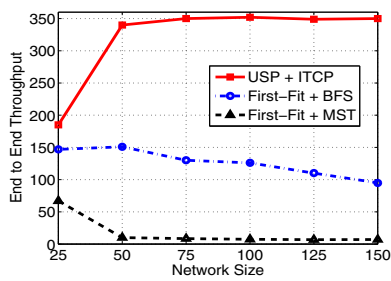


(b) Minimum satisfaction ratio

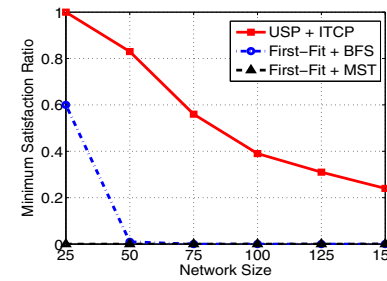


(c) Fairness index

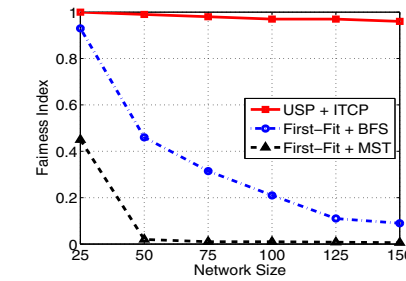
Fig. 5. The scheduling algorithms for the general case



(a) End-to-end throughput



(b) Minimum satisfaction ratio



(c) Fairness index

Fig. 6. The complete solutions

and “ITCP” represents our tree construction algorithm. We make the following observations from Figs. 3–6.

1) As shown in Fig. 3, compared to the BFS and MST algorithms, on average, our tree construction algorithm improves the minimum satisfaction ratio by 21% and 120%, the fairness index by 15% and 10%, and the end-to-end throughput by 3% and 140%, respectively. Essentially, more links in an end-to-end path (larger tree height) will normally lead to worse performance because it is more likely that allocating enough resources to an end-to-end path may fail. According to our observations, an MST usually has a larger height than the tree constructed by our routing algorithm. A BFS tree has smaller height than an MST. However, the BFS trees are usually not balanced, i.e., a particular node may have a relatively large number of descendants, which is obviously a negative factor for achieving good performance.

2) Our scheduling algorithms always perform the best in both the special and the general cases. Specifically, compared to the first-fit algorithm, our scheduling algorithm (for the general case) can significantly improve the end-to-end throughput. Moreover, no matter how large the network is, the fairness indices given by our scheduling algorithms are always very close to 1.0, which indicates that our algorithms can achieve a fair bandwidth allocation. As expected, the trivial algorithm performs very poorly in terms of both throughput and fairness, since it does not take advantage of spacial reuse.

3) Not surprisingly, the complete solution using our scheduling and routing algorithm significantly outperforms all other solutions. Specifically, compared to the first-fit+BFS solution, our solution achieves an average improvement of 154% on the end-to-end throughput, 446% on the minimum satisfaction ratio, and 177% on the fairness index.

4) A denser (larger) network usually has heavier traffic demands, and is supposed to result in higher throughput. Therefore, we can see from Figs. 3–6 that the end-to-end throughput given by our scheduling and routing algorithms always increases with the network size. However, more SAs introduce stronger interference, which will hold back the throughput improvement. Therefore, the throughput given by algorithms without carefully addressing interference such as the first-fit algorithm may even decrease with the network size. In addition, the minimum satisfaction ratio and the fairness index always decrease with the network size because it is more difficult to achieve good fairness in a larger network.

## VIII. CONCLUSIONS

In this paper, we studied routing and scheduling in wireless backhaul networks with smart antennas. We formally defined the Interference aware Tree Construction Problem (ITCP) for routing and presented a polynomial-time algorithm to solve it. It has been shown by simulations that the trees constructed by our algorithm outperform the well-known MST and BFS trees. We presented a polynomial-time optimal algorithm for a special case of the scheduling problem as well as an effective heuristic algorithm for the general case. Our simulation results showed that compared with other solutions such as the first-fit+BFS solution, our interference aware routing and scheduling scheme can improve throughput by 154% and fairness index by 177% on average.

## REFERENCES

- [1] IEEE 802.16 Working Group, Part 16: Air interface for fixed broadband wireless access systems, *IEEE Standard*, 2004.
- [2] R. Bhatia and L. Li, Throughput optimization of wireless mesh networks with MIMO links, *Proceedings of IEEE Infocom'2007*, pp. 2326–2330.
- [3] M. Cao, V. Raghunathan and P. R. Kumar, A tractable algorithm for fair and efficient uplink scheduling of multi-hop WiMax mesh networks, *Proceedings of IEEE WiMesh'2006*, pp. 101–108.
- [4] T. H. Cormen *et al.*, Introduction to algorithms, 2nd edition, *The MIT Press*, 2001.
- [5] R. R. Choudhury and Nitin H. Vaidya, Deafness: A MAC problem in ad hoc networks when using directional antennas, *Proceedings of IEEE ICNP'2004*, pp. 283–292.
- [6] C. Cicconetti, I. F. Akyildiz and Luciano Lenzini, Bandwidth balancing in multi-channel IEEE 802.16 wireless mesh networks, *Proceedings of IEEE Infocom'2007*.
- [7] S. Deb, V. Mhatre and V. Ramaiyan, WiMAX relay networks: opportunistic scheduling to exploit multiuser diversity and frequency selectivity, *Proceedings of MobiCom'2008*, pp. 163–174.
- [8] P. Gupta and P. R. Kumar, The capacity of wireless networks, *IEEE Transactions on Information Theory*, Vol. 46, No. 2, 2000, pp. 388–404.
- [9] M. Hu and J. Zhang, MIMO ad hoc networks: Medium Access Control, saturation throughput, and optimal hop distance, *Journal of Communications and Networks*, Special Issue on Mobile Ad Hoc Networks, 2004, pp. 317–330.
- [10] R. Jain, D. M. Chiu and W. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared systems, *DEC Research Report TR-301*, 1984.
- [11] K. Jain, J. Padhye, V. Padmanabhan, and L. Qiu, Impact on interference on multihop wireless network performance, *Proceedings of ACM MobiCom'2003*, pp. 66–80.
- [12] A. Kabbani, T. Salonidis, and E. Knightly, Distributed low-complexity maximum-throughput scheduling for wireless backhaul networks, *Proceedings of IEEE Infocom'2007*, pp. 2063–2071.
- [13] D. Kim and A. Ganz, Fair and efficient multihop scheduling algorithm for IEEE 802.16 BWA systems, *Proceedings of IEEE Broadnets'2005*, pp. 833–839.
- [14] T. Korakis, G. Jakllari and L. Tassioulas, A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks *Proceedings of ACM MobiHoc'2003*, pp. 98–107.
- [15] Y. Lin, T. Javidi, R. L. Cruz and L. B. Milstein, Distributed link scheduling, power control and routing for multi-hop wireless MIMO networks, *Proceedings of IEEE Asilomar Conference on Signals, Systems, and Computers*, 2006, pp. 122–126.
- [16] B. Mumei, J. Tang and T. Hahn, Joint stream control and scheduling in multihop wireless networks with MIMO links, *Proceedings of IEEE ICC'2008*, pp. 2921–2925.
- [17] S. Ramanathan, A unified framework and algorithm for channel assignment in wireless networks, *Kluwer/ACM Journal of Wireless Networks*, Vol. 5, No. 2, 1999, pp. 81–94.
- [18] H. Shetiya and V. Sharma, Algorithms for routing and centralized scheduling to provide QoS in IEEE 802.16 mesh networks, *Proceedings of ACM workshop on Wireless multimedia networking and performance modeling*, 2005, pp. 140–149.
- [19] K. Sundaresan, R. Sivakumar, M. A. Ingram and T-Y. Chang, Medium access control in ad hoc networks with MIMO links: optimization considerations and algorithms, *IEEE Transactions on Mobile Computing*, Vol. 3, No. 4, 2004, pp. 350–365.
- [20] K. Sundaresan and R. Sivakumar, A unified MAC layer framework for ad-hoc networks with smart antennas, *Proceedings of ACM MobiHoc'2004*, pp. 244–255.
- [21] K. Sundaresan, W. Wang and S. Eidenbenz, Algorithmic aspects of communication in ad hoc networks with smart antennas, *Proceedings of ACM MobiHoc'2006*, pp. 299–309.
- [22] K. Sundaresan and S. Rangarajan, On exploiting diversity and spatial reuse in relay-enabled wireless networks, *Proceedings of ACM MobiHoc'2008*, pp. 13–22.
- [23] J. Tao, F. Liu, Z. Zeng and Z. Lin, Throughput enhancement in WiMax mesh networks using concurrent transmission, *IEEE International Conference on Wireless Communications, Networking and Mobile Computing*, 2005, pp. 871–874.
- [24] H. Y. Wei, S. Ganguly, R. Izmailov, and Z. Haas, Interference-aware IEEE 802.16 WiMax mesh networks, *Proceedings of IEEE VTC'2005*, pp. 3102–3106.