INTERFERENCE AWARE ROUTING AND SCHEDULING IN WIMAX BACKHAUL NETWORKS WITH SMART ANTENNAS

by

Shen Wan

A project report submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

November 2008

APPROVAL

of a project report submitted by

Shen Wan

This project report has been read by each member of the committee and has been found to be satisfactory regarding content, English usage, format, citations, bibliographic style, and consistency, and is ready for submission to the College of Graduate Studies.

Dr. Jian (Neil) Tang

Approved for the Department of Computer Science

Dr. John Paxton

Approved for the College of Graduate Studies

Dr. Robert Marley

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

GLOSSARY

802.11 — IEEE 802.11 is a set of standards for wireless local area network (WLAN) computer communication in the 5 GHz and 2.4 GHz public spectrum bands.

802.16 — The IEEE WiMAX standard set.

802.16j — An amendment to the IEEE 802.16 standard, focused on multihop relay specification.

BS (Base Station) — A radio transceiver that serves as the hub of the WiMAX network and the gateway to the external network, such as the Internet. In centralized scheme, BS is in charge of allocating the WiMAX network resource and coordinate the communication of other SSs.

diversity gain — The performance improvement of communnication resulted by using different time, frequency, path, etc to transmit the same message.

DOF (Degrees of Freedom) — The number of beams/nulls that a smart antenna can provide, usually equal to the number of elements in the array.

effective satisfaction ratio — The minimum satisfaction ratio of all descendants of a node.

smart antenna — Also known as adaptive array antennas, a type of antenna which can adapt its transceiving to the environment, forming beams/nulls toward directions to enhance signal strength and supress interference.

MIMO (Multiple Input Multiple Output) antenna — A technology to use multiple antennas at both the transmitter and receiver to improve the communication performance. It is one form of smart antenna technology.

primary interference — When two unicast communication links are incident to each other (share a common node), they cannot be active simultaneously using half-duplex transceivers. This is defined as *primary interference* between these two links.

satisfaction ratio — The ratio of the bandwidth allocated to a node to the bandwidth request of the node.

secondary interference — When two links do not have primary interference in between, but they cannot be active simultaneously because at least one link would have Signal-to-Interference-and-Noise Ratio (SINR) dropped below the threshold, it is said to have *secondary interference* between these two links.

SS (Subscriber Station) — A radio transceiver that serves as the interface to WiMAX network for a client/subscriber. The client/subscriber may be a device or a local area network. SS may communicate directly to the BS or use other SSs as relay in a multihop fashion.

TDMA (Time Division Multiple Access) — A method for multiple users to share a channel. The signal is divided to different time slots. Each user will only access the channel in his/her own time slots.

WiMAX — An acronym for *Worldwide Interoperability for Microwave Access*, is described by the WiMAX forum as "a standards-based technology enabling the delivery of last mile wireless broadband access as an alternative to cable and DSL". It has been standardized as IEEE 802.16.

ABSTRACT

A smart adaptive antenna can provide multiple Degrees of Freedom (DOFs), which can be used for intended communications and interference suppression. By incorporating smart antennas in a WiMAX system, network throughput can be significantly improved by more efficient spatial reuse. In this project, we consider routing and scheduling in WiMAX backhaul networks with smart antennas. We formally define the Interference-aware Tree Construction Problem (ITCP) for routing, which offers full consideration for interference impact and DOF availability. We then present an algorithm to optimally solve it in polynomial time. As for scheduling, we first present a polynomial-time, optimal algorithm for a special case in which the number of DOFs in each node is large enough to suppress all potential secondary interference. An effective heuristic algorithm is then presented for the scheduling problem in the general case. Extensive simulations are conducted to evaluate the performance of the proposed algorithms.

INTRODUCTION

Background

The emerging WiMAX technology (IEEE 802.16 [1]) can offer low-cost, high-speed and long-range communications for applications ranging from broadband Internet access to military and emergency communications.

A WiMAX network is composed of a Base Station (BS) and multiple Subscriber Stations (SS). The BS serves as a gateway connecting the WiMAX network to external networks such as the Internet. Two operation modes are supported by the standard: Point-to-Multipoint (PMP) mode and mesh mode.

Quite similar to a cellular network, a WiMAX network working in the PMP mode is essentially a single-hop wireless network in which an SS always directly communicates with the BS. In mesh mode, a spanning tree rooted at the BS is formed for routing. An SS out of the transmission range of the BS can use other SSs as relay to communicate with the BS in a multihop fashion. Compared to the PMP mode, the mesh mode can significantly extend wireless coverage and improve network capacity. A WiMAX backhaul network is illustrated in Fig. 1.

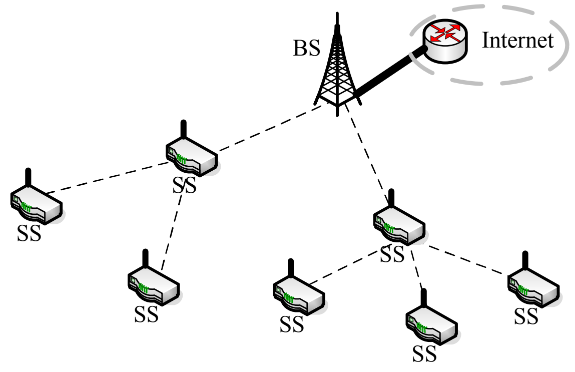


Figure 1. A WiMAX backhaul network.

Unlike a conventional omni-directional antenna which wastes most of its energy in directions where there is no intended receiver, a smart (directional) antenna offers a longer transmission range and lower power consumption by forming one or multiple beams towards intended receivers only. The emerging Digital Adaptive Array (DAA) antennas [2] can even perform fine-grained interference suppression by adaptively forming nulls in certain directions using its antenna elements (a.k.a, Degrees Of Freedom (DOFs)), which leads to better spatial utilization. Therefore, smart antennas can enhance the functionalities of a WiMAX system and help it better achieve the goal of providing long-range and high-speed communications. The detailed antenna operation model will be discussed in Chapter 3. Although DAA antennas have been extensively studied before, research on backhaul networks using DAA antennas is still in its infancy.

Our Work and Contributions

We consider two fundamental problems, routing and scheduling, in WiMAX backhaul networks with smart antennas. The WiMAX standard [1] specifies a common MAC protocol for both the PMP mode and the mesh mode, including signaling protocols and message structures. But the standard does not specify either the algorithm for computing transmission schedule (i.e., scheduling algorithm) or the routing protocol. The routing problem, i.e., the tree construction problem, has not been well studied for WiMAX backhaul networks, especially for those with DAA antennas. However, without careful consideration for interference impact and resource availability, scheduling transmissions along the constructed tree may lead to poor throughput and serious unfairness. In the simulation, we show that a Minimum Spanning Tree (MST) based routing approach performs very poorly.

Scheduling links with DAA antennas is quite different from traditional link scheduling with omni-directional antennas since simultaneous transmissions on two interfering links can be supported as long as DOFs are properly assigned to suppress interferences. The scheduling problem involves both link scheduling and DOF assignment, which makes it more challenging compared to scheduling with omni-directional antennas. To our best knowledge, we are the first to address routing and scheduling problems in the context of WiMAX backhaul networks with DAA antennas. Our contributions are summarized in the following.

1. For routing, we formally define the Interference-aware Tree Construction Problem (ITCP), which offers full consideration for interference impacts and DOF availability. We present an algorithm to optimally solve it in polynomial time.

2. We consider a special case of the scheduling problem where the number of DOFs in each node is large enough to suppress all *potential secondary interference.* We present a polynomial-time optimal algorithm to solve it.

3. We present an effective heuristic algorithm for the general case of the scheduling problem, whose performance is justified by extensive simulations.

The rest of this report is organized as follows. We discuss related works in Chapter 2. We describe the system model and formally define the problems in Chapters 3 and 4, respectively. The tree construction algorithm is presented on Chapter 5. We present our scheduling algorithms in Chapter 6. The simulation results are presented in Chapter 7 and the report is concluded in Chapter 8.

RELATED WORK

Previous Work on WiMAX Scheduling

Transmission scheduling in WiMAX backhaul networks with omni-directional antennas has been studied recently. Different centralized heuristic algorithms have been proposed for scheduling and/or routing in [3, 4, 5, 6] with the objective of maximizing spatial reuse.

In [7], Cao *et al.* introduced a new fairness notion that is imposed contingent on the actual traffic demands. They presented an optimal algorithm to solve a scheduling problem whose objective is to maximize network throughput within their fairness model.

In [8], the authors focused on Quality of Service (QoS) support and proposed routing and scheduling algorithms to provide per-flow QoS guarantees.

In [9], a distributed algorithm was presented to provide fair end-to-end bandwidth allocation for single-radio, multi-channel WiMAX backhaul networks.

In [10], Sundaresan *et al.* showed that the scheduling problem to exploit diversity gains alone in a 2-hop 802.16*j*-based backhaul network is NP-hard. They then provided polynomial-time approximation algorithms. They also proposed a heuristic algorithm to exploit both spatial reuse and diversity.

The general link scheduling for multihop wireless networks with omni-directional antennas has also been studied in [11, 12].

Previous Work on Smart Antennas

Smart antennas have also received tremendous attention due to their capabilities of range extension, power saving and interference suppression.

MAC protocols have been proposed in [13, 14] for 802.11-based ad-hoc networks using either switched beam antennas or adaptive antennas. The authors of those papers modified the original 802.11 MAC protocol to exploit the benefits of smart antennas.

In a recent work [2], the authors considered the problem of determining DOF assignment for DAAs with the objective of interference minimization. They presented constant factor approximation algorithms to solve it. Moreover, they proposed a distributed algorithm for joint DOF assignment and scheduling.

Another important type of smart antenna is Multiple Input Multiple Output (MIMO) antenna which is able to support multiple concurrent streams over a single link.

In [15], the authors discussed key optimization considerations, such as spatial multiplexing, for MAC layer design in ad-hoc networks with MIMO links. They presented a centralized algorithm as well as a distributed protocol for stream control and medium access with those key optimization considerations incorporated.

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 [17].

In [18], Hu and Zhang examined the impact of spatial diversity on the MAC design, and devised a MIMO MAC protocol accordingly. They also studied the impact of MIMO MAC on routing and characterized the optimal hop distance that minimizes the end-to-end delay in a large network.

Cross-layer optimization for MIMO-based wireless networks has also been studied in [19, 20]. In [19], Bhatia and Li presented a centralized algorithm to solve the joint routing, scheduling and stream control problem subject to fairness constraints.

Difference of Our Work to Previous Works

The difference between our work and these related works is summarized as follows:

1. As mentioned before, due to the interference suppression feature of DAA antennas, the scheduling problem studied here is significantly different from the scheduling problems with omni-directional or MIMO based antennas.

2. Our optimization goal is to improve end-to-end throughput and fairness. However, the general scheduling problems studied in [21, 14, 16, 15, 17, 2] aim at maximizing single-hop throughput or minimizing the frame length.

3. The routing tree and transmission schedule computed by our algorithms have certain good, provable properties which cannot be supported by the heuristic algorithms reported in [9, 3, 4, 8, 5, 6].

SYSTEM MODEL

In this chapter, we describe our models and assumptions for antenna operations, networking and interference.

Antenna Model

Similar as in [2], we focus on DAA antenna, which is a type of well-known smart antennas. A DDA with K elements is said to have K DOFs. In order to enable communications between a pair of transmitter and receiver (link) (v_i, v_j), a single DOF needs to be assigned for communications at each end. In this way, the Signal-to-Noise-Ratio (SNR) can be improved at the receiver, and therefore the transmission range can be increased. Moreover, except for the DOF assigned for communications, the remaining $(K-1)$ DOFs can be used at the transmitter to cancel its interference to other receivers by forming nulls at corresponding directions. Similarly, the receiver can use its remaining $(K-1)$ DOFs to suppress interference from other transmitters. At directions where neither beam of null has been assigned, communication is impossible but there may still exist interference.

Scheduling Frame Model

The WiMAX standard [1] adopts a Time Division Multiple Access (TDMA) 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[1] subframe and a downlink[2] subframe with T^u and T^d minislots respectively. Unlike the PMP mode, T^u does not have to be the same as T^d in the mesh mode. In some systems, how to

[1]the direction from SS to BS
[2]the direction from BS to SS

split T^d and T^u are predetermined and we simply follow the specification. Otherwise, we consider the network with both downlink and uplink demands, use Algorithm 2 in Chapter 6 to find the bottleneck node. If there are Q^d downlink demands and Q^u uplink demands at the bottleneck node, we split downlink subframe and uplink subframe in the same proportion: $T^d = \frac{Q^d T}{Q^d + Q^u}$ and $T^u = \frac{Q^u T}{Q^d + Q^u}$. In this way, downlink demands and uplink demands will have the same minimum satisfaction ratio.

The WiMAX MAC protocol [1] supports both centralized and distributed scheduling. In this project, we focus on centralized scheduling, which is composed of two phases. In the first phase, each SS transmits an MSH-CSCH Request (Mesh Centralized Scheduling) message carrying bandwidth request information to its parent node in the routing tree. Each non-leaf SS also needs to include bandwidth requests from its children in its own request message. In the second phase, the BS determines the bandwidth allocation for each SS based on all requests collected in the first phase and notify SSs by broadcasting an MSH-CSCH Grant message along the routing tree. Subsequently, each SS computes the actual transmission schedule based on the bandwidth granted by the BS using a common scheduling algorithm. The time period for exercising these two stages is referred to as a scheduling period whose duration is a multiple of the frame duration, depending on the network size. We assume that the bandwidth requests do not change within a scheduling period.

Network Model

We consider a static WiMAX backhaul network with one Base Station (BS) and $(n-1)$ Subscriber Stations (SSs). These nodes will form a spanning tree rooted at the BS for routing. An SS may need to use other SSs as relay to communicate with the BS in a multihop fashion. Each node has a single transceiver and a DAA antenna with K DOFs. All transmissions are conducted on a single common channel. We model the network using a *communication graph* $G(V, E)$, where $V = \{v_0, v_1, \ldots, v_{n-1}\}$. v_0 corresponds to the BS, whereas $v_1, \ldots, v_{n-1}$ are SSs. If all nodes are placed on a plane with no blockage in between and transmit using the same fixed power, 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 with the same power level. In this case, $(v_i, v_j) \in E$ if $\|v_i - v_j\| \leq R_T$.[3] 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 [2]. 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 interference from 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 and normally 2–3 times larger than R_T. If two links $e = (v_i, v_j)$ and $e' = (v_{i'}, v_{j'})$ are incident, we say there exists *primary interference* in between. In this case, in no way can they 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 the same 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 $v_{j'} \in N_i$, we say there exists *secondary interference* in between. A DOF can be assigned at either the transmitter v_i or the receiver $v_{j'}$ to suppress secondary interference from e to e'. Similarly, a DOF can be assigned at either the transmitter $v_{i'}$ or the receiver v_j to suppress secondary interference from e' to e.

[3] $\|v_i - v_j\|$ indicates the Euclidean distance between node v_i and v_j.

PROBLEM DEFINITION

In this chapter, we formally define the problems to study. All the related notations are summarized in Table 1.

Table 1. 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
m/n	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
$\mathbf{\Gamma}$	The scheduling matrix
$\mathbf{\Lambda}$	The DOF assignment matrix

Routing Problem

First of all, we define the routing problem since a routing tree is given as the input for a scheduling problem. It would be more precise to tell which tree is the best with bandwidth demand information. However, a routing tree is normally constructed beforehand and will be used for a relatively long period during which the traffic demands may change.

It is well-known that interference has significant impacts on network performance [22]. So we will construct a low-interference tree, which can hopefully 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. Therefore, once the communication graph G is given, we can easily determine on which layer of the routing tree should an SS v_i 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 (not including v_0), i.e., the height of the tree. The tree construction problem is essentially the problem to determine 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 the primary and secondary interference because the primary interference can only be resolved by scheduling but the 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 $h > 1$, $I^p_{\max} = \max_{v \in V_h} I^p(v)$ 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 possible to be eliminated by properly assigning DOFs.[1]

Scheduling Problem

The scheduling problem considered here is different from that in a network with omni-directional antennas because DOFs can be used to suppress interference. Therefore, our scheduling problems involve both transmission scheduling and DOF assignment. The inputs of the scheduling problem include a spanning tree with the BS v_0 as the root, $(n-1)$ SSs $\{v_1, \ldots, v_{n-1}\}$, their bandwidth demands $\mathbf{Q^u} = [q_1^u, \ldots, q_{n-1}^u]$ and $\mathbf{Q^d} = [q_1^d, \ldots, q_{n-1}^d]$ for uplink and downlink respectively, and the uplink/downlink subframe sizes T^u/T^d. q_i^u indicates 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 $\mathbf{\Gamma}$ and a DOF assignment matrix $\mathbf{\Lambda}$. $\mathbf{\Gamma}_i^t = 1$ if link (v_i, v_{p_i}) is active in minislot t; $\mathbf{\Gamma}_i^t = 0$ otherwise.[2] $\mathbf{\Lambda}_{i,j}^t = 1$ if v_i assigns a DOF to point at v_j for communications or interference supression in minislot t; $\mathbf{\Lambda}_{i,j}^t = 0$ otherwise.

A scheduling matrix $\mathbf{\Gamma}$ and a DOF assignment matrix $\mathbf{\Lambda}$ are said to be *feasible* if:

1. $\forall i, t, \mathbf{\Lambda}_{i,p_i}^t \cdot \mathbf{\Lambda}_{p_i,i}^t \geq \mathbf{\Gamma}_i^t$ (for each active link, DOFs need to be assigned at both end for communication);

2. there does not exist primary or secondary interference in any minislot;

3. $\forall i, t, \sum_{j=0}^{n-1} \mathbf{\Lambda}_{i,j}^t \leq K$ (each node has only K DOFs).

We also define an uplink bandwidth allocation vector $\mathbf{B^u} = [b_1^u, \ldots, b_{n-1}^u]$, its corresponding satisfaction ratio vector $\mathbf{S^u} = [s_1^u, \ldots, s_{n-1}^u] = [\frac{b_1^u}{q_1^u}, \ldots, \frac{b_{n-1}^u}{q_{n-1}^u}]$,[3] and its

[1]Check Algorithm 6 in Chapter 6.

[2]Only $(n-1)$ links on the given tree will be considered for scheduling.

[3]We define the satisfaction ratio of $\frac{0}{0} = 1$.

corresponding aggregated bandwidth allocation vector $\mathbf{A^u} = [a_1^u, \ldots, a_{n-1}^u]$, where b_i^u indicates the actual bandwidth (the number of minimslots 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 $\mathbf{\Gamma}$, such that $\forall i \in \{1, \ldots, n-1\}, \sum_{t=1}^{T^u} \mathbf{\Gamma}_i^t \geq a_i^u$.

Based on a scheduling matrix $\mathbf{\Gamma}$, the sustainable data rate on link $e_i = (v_i, v_{p_i})$ is $b(e_i) = \sum_{t=1}^{T^u} \mathbf{\Gamma}_i^t$.

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 $\mathbf{\Gamma}$ and DOF assignment matrix $\mathbf{\Lambda}$ 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 fairness purpose. Similarly, we can define the Downlink Scheduling Problem (DSP). Note that if there exists an algorithm which can optimally solve the USP/DSP, then it can tell if a bandwidth demand vector can be fully satisfied ($\min_{1 \leq i \leq n-1} s_i^u = 1$) or not ($\min_{1 \leq i \leq n-1} s_i^u < 1$).

THE TREE CONSTRUCTION ALGORITHM

We present Algorithm 1, an optimal algorithm, to solve the ITCP. Its inputs include the communication graph G, an array H which gives the layer of each node, and $h_{\max}$, the number of layers.[1] An array *Parent* is the output, which specifies the parent node for each SS.

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

```
Step 1  $V_1 \leftarrow \{v | H[v] = 1\}$;
        forall $v \in V_1$ $Parent[v] \leftarrow v_0$; endforall
        forall $v \in V \setminus V_1$ $Parent[v] \leftarrow$ null; endforall
        $h \leftarrow h_{\max}$;

Step 2  if $(h = 1)$ return $Parent$;

Step 3  $V_h \leftarrow \{v | H[v] = h\}$;
        $V_{h-1} \leftarrow \{v | H[v] = h - 1\}$;
        $d_{\max} \leftarrow \max_{v \in V_{h-1}} d_v$, 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 on $G'$ to find the maximum flow $f$ from
        node $s$ to node $t$ and the corresponding link flow allocation $Flow$;
          if $(f = |V_h|)$ $ub \leftarrow mid - 1$;
          else $lb \leftarrow mid + 1$;
          endif
        endwhile

Step 5  forall $e = (u, v) \in E'$
          if $(Flow[e] = 1$ and $u \neq s$ and $v \neq t)$
            $Parent[u] \leftarrow v$;
          endif
        endforall

Step 6  $h \leftarrow h - 1$;
        goto Step 2;
```

[1] G is treated as an undirected graph for the tree construction. H and $h_{\max}$ can be computed using Breadth-First-Search.

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, each node in layer h is connected to some node in layer $(h-1)$, while the maximum degree of nodes in layer $(h-1)$ is minimized using binary search. The auxiliary graph $G'(V', E')$ in Step 4 is constructed as follows: $V' = V_h \cup V_{h-1} \cup \{s, t\}$, where s and t are virtual source and sink nodes respectively. For each $v \in V_h$, we create a directed link with a capacity of 1 from s to v. For each $u \in V_h$ and $v \in V_{h-1}$, we create a directed link e with capacity 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 t. 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. **??**. 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 more than the corresponding bound $I^b(h)$.

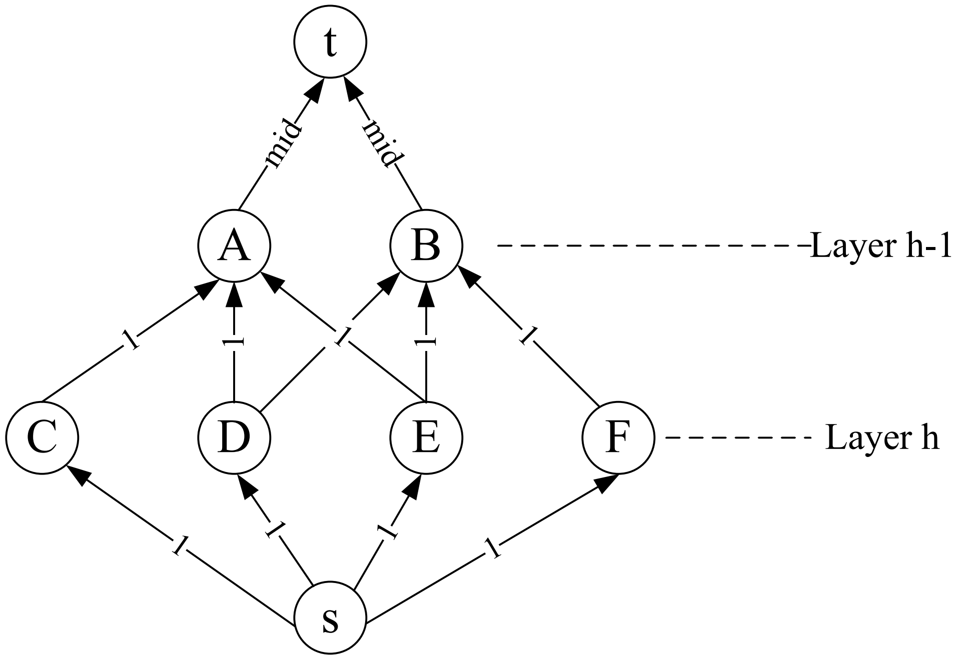


Figure 2. The auxiliary graph G'.

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

Proof: As mentioned before, the ITCP problem is essentially the problem to determine 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 boils down 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 degree of nodes in V_{h-1} is minimized. Note that the primary interference value of a non-leaf node is actually equal to its degree. In each iteration of the binary search in Step 4, we need to check if there exits an assignment such that each node in V_h is connected to exactly one node in V_{h-1} and the degrees of each node in V_{h-1} is no more than *mid*. Next, we show that there exists such an assignment if and only if the maximum s–t flow is equal to $|V_h|$.

First of all, it is well-known that the augmenting path based maximum flow algorithm such as 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. 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 $v \in V_h$ is 1. According to the flow conservation constraint and 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 t are set to *mid*, which ensures that based on the assignment, 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. 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, $d_{\max}$ is bounded by $(\delta_{\max}-1)$, where $\delta_{\max}$ is the maximum node degree in the communication graph G. 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 $O(|E'|) = O(m)$ 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 tree construction algorithm runs very fast in practice because the number of links between two consecutive layers are usually much less than m, and $h_{\max}$ and $\delta_{\max}$ are normally small.

THE SCHEDULING ALGORITHMS

In this chapter, we will present algorithms to solve the scheduling problems. Since the uplink and downlink traffic are scheduled in different subframes according to the WiMAX MAC protocol [1], *we will 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 [11, 12]. 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 in the active receivers and another half can be dealt with by DOFs in the active transmitters. Therefore, in this special case, only the impact of 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 the primary and secondary interference need to be addressed.

The Scheduling Algorithm for the Special Case

In the special case, a bandwidth allocation vector **B** is feasible if for its corresponding aggregated bandwidth allocation vector **A**,

$$\forall v, \sum_{i \in D_v} a_i \leq T \text{ ,where } D_v = \{i | v_i \text{ is a descendant of } v \text{ or } v \text{ itself}\}. \tag{6.1}$$

Therefore, it suffices to find a bandwidth allocation vector **B** with constraints in (6.1).

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. This procedure will be repeated until all the nodes are removed from the tree. The algorithm for solving the special case USP is formally presented as Algorithm 2, whose inputs include the bandwidth demand vector $\mathbf{Q^u} = [q_1^u, ..., q_{n-1}^u]$, the number of minislots available for uplink traffic T^u and the routing tree Y.

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

```
Step 1  $P \leftarrow \{i | v_i$ is a non-leaf node on $Y\}$;
        $Z \leftarrow \{0, 1, \cdots, n-1\}$
        forall $i \in P$ $T_i \leftarrow T^u$; endforall

Step 2  $\gamma_0 \leftarrow$ Schedule-BS($\mathbf{Q^u}, T_0$);
        forall $i \in P \setminus \{0\}$
           $\gamma_i \leftarrow$ Schedule-SS($\mathbf{Q^u}, T_i, i$);
        endforall
        $j \leftarrow \text{argmin}_{i \in P} \gamma_i$;

Step 3  $D \leftarrow \{i | v_i$ is a descendant of $v_j$ on $Y\}$;
        $C \leftarrow \{i | v_i$ is an ancestor of $v_j$ on $Y\}$;
        $B_j \leftarrow \sum_{i \in D} b_i$;
        forall $i \in C$ $T_i \leftarrow T_i - B_j$; endforall
        $Z \leftarrow Z \setminus \{j\} \setminus D$;
        if $Z \neq \emptyset$ goto Step 2; endif
```

In Step 1 of Algorithm 2, we initiate the number of free minislots at all non-leaf nodes to T^u.[1] The algorithm 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 will be discussed later. In Step 3, the bottleneck node and all of its descendants are removed from the tree, and the numbers of free minislots in all of its ancestors are updated. The procedure is repeated until all nodes are scheduled.

Since for all nodes other than the bottleneck node v_b, we have found scheduling schemes in Step 2 such that $\forall k, \gamma_k \geq \gamma_b$. We guarantee that other bandwidth demands scheduled later will have satisfaction ratios greater than or equal to γ_b.

[1]There is no need to consider any leaf node, since the scheduling scheme for its parent node contains the minislot allocation for the link in between, which is the only link incident to the leaf node.

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

```
Step 1  $\mathbf{B} \leftarrow \mathbf{0}$; $q_{\text{total}} \leftarrow \sum_{1 \le i \le n-1} q_i$;
        **if** $q_{\text{total}} \le T_0$
            $\mathbf{B} \leftarrow \mathbf{Q}$; **return** 1;
        **endif**

Step 2  $\gamma \leftarrow \frac{T_0}{q_{\text{total}}}$;
        **forall** $k \in \{1, \ldots, n-1\}$
            $b_k \leftarrow \lfloor \gamma q_k \rfloor$; $s_k \leftarrow \frac{b_k}{q_k}$; ($s_k \leftarrow 1$ if $q_k = 0$)
        **endforall**
        $T_{\text{rem}} \leftarrow T_0 - \sum_{1 \le k \le n-1} b_k$;

Step 3  $j \leftarrow \text{argmin}_{1 \le k \le n-1}\, s_k$;
        **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  $\mathbf{B} \leftarrow \mathbf{0}$; $D \leftarrow \{j | v_j$ is a descendant of $v_i$ on $Y\}$;
        $D' \leftarrow D \cup \{i\}$;
        $q_{\text{total}} \leftarrow q_i + 2\sum_{k \in D} q_k$;
        **if** $q_{\text{total}} \le T_i$
            **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.[2] In both algorithms, q_{total} gives the total number of minislots required for the traffic that needs to go through the corresponding node.[3] 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 at v_i. 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) will be allocated to nodes in 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 will be identified as the bottleneck node.[4] 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 the child to relay v_k's traffic and the parent node of v_i respectively. However, the bandwidth allocation in the BS is simpler since it does not have any parent node.

Algorithm 3 seeks a scheduling scheme at root node, updates the bandwidth allocation vector $\mathbf{B}$, and returns the minimum satisfaction ratio of all demands. In Step 1, we sum up all the demands and satisfy all demands if the available minislots are enough. Otherwise, we scale down all demands by γ and count the remaining free minislots N in Step 2. Then we keep allocating one minislot to the demand

[2]Once the bandwidth allocation is determined, it is trivial to find a corresponding transmission schedule and DOF assignment in the special case.

[3]Including the traffic generated by itself and all its descendants.

[4]The number of minislots that can be allocated for a node v_k at different its ancestors may be different.

with lowest satisfaction ratio in Step 3 until all free minislots are depleted. Similar to Algorithm 3, Algorithm 4 seeks a bandwidth allocation at an SS v_i with available minislots T_i and bandwidth demands $\mathbf{Q}$, updates part of the bandwidth allocation vector $\mathbf{B}$, and returns the effective satisfaction ratio of all demands initiated from the subtree rooted at this intermediate node.

We will use an example to demonstrate how Algorithms 2, 3 and 4 solve the scheduling problem in the special case. The example network is shown in Fig. 3.

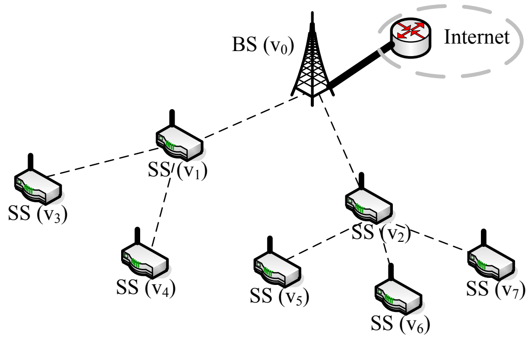


Figure 3. An example WiMAX network.

In Fig. 3, we denote the BS as v_0, the two relay SSs as v_1 and v_2, and the leaf level SSs $v_3, \ldots, v_7$ (all from left to right). Let bandwidth requests of each SS be $\mathbf{Q} = [1, 1, 2, 3, 2, 3, 4]$, and the number of minislots in a scheduling period be $T = 16$.

First, we need to find out the bottleneck node in the network, which has the lowest effective satisfaction ratio. We accumulate all bandwidth requests through v_0 and get $Q_{\text{total}} = 16 \leq T$. This shows that all requests can be satisfied from v_0's point of view. And therefore its effective satisfaction ratio is 100%. Similarly, we pre-schedule for v_1 and find out v_1 is not the bottleneck node either. When we then pre-schedule for v_2, we get $Q_{\text{total}} = 19 > T$. This shows that not all requests through v_2 can be satisfied. We first allocate $\frac{16}{19}$ of the requested minislots to v_2, v_5, v_6 and v_7, rounding down if needed. Accordingly, the four nodes get 0, 1, 2 and 3 minislots with satisfaction ratios 0, $\frac{1}{2}$, $\frac{2}{3}$ and $\frac{3}{4}$ respectively, and we have 4 free minislots left. Then we allocate those 4 minislots, starting from the most unsatisfied node v_2, then v_5. Note that in order to allocate one more minislot to v_5, two minislots need to be consumed at v_2 because v_2 needs to receive data from v_5 and then forward them to v_0. Therefore, after allocating minislots to v_5, there is only one minislot left and no more allocation for v_6 or v_7 is possible. We end up with the four nodes each allocated 1, 2, 2 and 3 minislots, with satisfaction ratios 1, 1, $\frac{2}{3}$ and $\frac{3}{4}$. The effective satisfaction ratio for v_2 is $\frac{2}{3}$, the lowest of the four ratios. Therefore, v_2 is the bottleneck node.

After scheduling for all requests through v_2, we remove the subtree rooted at v_2 and all its requests from the graph, locate the next bottleneck node and schedule for the remaining tree recursively. In this example, the remaining requests can all be fully satisfied. The final allocation vector for the entire graph is $\mathbf{B} = [1, 1, 2, 3, 2, 2, 3]$. Except for nodes v_6 and v_7, all nodes' bandwidth requests are satisfied.

Theorem 2: Algorithm 2 computes a bandwidth allocation vector $\mathbf{B}$ with max-min satisfaction ratio in $O(n^3 \log n)$ time.[5]

Proof: First of all, we show that Algorithm 3 always computes a bandwidth allocation vector $\mathbf{B}$ with max-min satisfaction ratio based on the given minislot availability. 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 is no free minislots left, 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

$$s'_j = \frac{b'_j}{q_j} \leq \frac{b_j - 1}{q_j} = \frac{\lfloor \gamma q_j \rfloor - 1}{q_j} \leq \gamma - \frac{1}{q_j} \tag{6.2}$$

In Step 2 of Algorithm 3, since each b_k is rounded down to $\lfloor \gamma q_k \rfloor$, $\forall k, (\gamma - s_{\min}) q_k \leq 1$. Therefore, $\gamma - \frac{1}{q_j} \leq s_{\min}$. Combining this with (6.2), we have $s'_j \leq s_{\min}$. This

[5]The input for scheduling problems are trees. Therefore $O(m) = O(n)$.

contradicts with 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., $s'_{\min} > s_{\min}$. Similarly, we can prove that Algorithm 4 also computes a bandwidth allocation vector $\mathbf{B}$ with max-min satisfaction ratio.

In Step 2 of Algorithm 2, if BS is the bottleneck node, i.e., $j = 0$, we will simply get the bandwidth allocation vector from Algorithm 3, which has been shown above to have the max-min satisfaction ratio. Next, we consider the case when the bottleneck node v_j is an SS, i.e., $j \neq 0$. Let $s_{\min} = \min_{1 \leq i \leq n-1} s_i$ be the minimum satisfaction ratio found by Algorithm 2. Since v_j is the bottleneck node, $s_{\min} = s_j$. 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 $\mathbf{B}'$: $s'_j \geq s'_{\min} > s_{\min} = s_j$. And this contradicts with our proof that s_j is maximized for the subtree rooted at v_j.

Next, we show Algorithm 2 is a polynomial-time algorithm. 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 $O(n^2 \log n)$. It takes $O(n)$ time for updating in Step 3. Step 3 removes at least one node from the spanning tree Y. The loop composed of Step 2 and Step 3 will be executed for $O(n)$ 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. Therefore, Step 2 of Algorithm 2 will only be executed a few times. And the running time of Algorithm 2 is only $O(n^2 \log n)$ in practice.

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  $\mathbf{\Gamma} \leftarrow \mathbf{0}$; $\mathbf{\Lambda} \leftarrow \mathbf{0}$; $\mathbf{B} \leftarrow \mathbf{0}$; $t \leftarrow 1$;
        **forall** $i \in \{1, \cdots, n-1\}$
          $a_i \leftarrow \Sigma_{k \in D_i} q_k + b_i$; $x_i \leftarrow 0$;
        **endforall**

Step 2  Sort $L^s$ in the ascending order of link satisfaction ratios;
        $L \leftarrow \emptyset$;

Step 3  **forall** $e = (v_i, v_j) \in L^s$
          $A \leftarrow$ AssignDOF$(L, e)$;
          **if** $(A \neq \emptyset)$
            $L \leftarrow L \cup \{e\}$; $\mathbf{\Gamma}_i^t \leftarrow 1$;
            $b_i \leftarrow b_i + 1$; $x_i \leftarrow x_i + 1$;
            **if** $(x_i = a_i)$ $L^s \leftarrow L^s \setminus \{e\}$; **endif**
            **forall** $(k, l) \in A$ $\mathbf{\Lambda}_{k,l}^t \leftarrow 1$; **endforall**
          **endif**
        **endforall**

Step 4  $t \leftarrow t + 1$;
        **if** $t \leq T^u$ **goto** Step 2;
```

Algorithm 5 computes a scheduling matrix $\mathbf{\Gamma}$ with a scheduling period of T^u minislots, a DOF assignment $\mathbf{\Lambda}$ and a corresponding bandwidth allocation vector $\mathbf{B}$ 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 which tries to pack as many links as possible in a minislot in ascending order of their satisfaction ratios. Note that the satisfaction ratio of a link is equal to the number of minislots which 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*.

In essence, Algorithm 5 greedily tries to maximize the minimum satisfaction ratio of all aggregated bandwidth demands. The algorithm activates individual links in each minislot instead of allocating minislots to individual bandwidth demands, which may need multi-hop transmissions. Therefore, it is possible for the algorithm to activate a link before activating links of its ancestors. If a link has less minislots allocated than all minislots allocated to its children links, then some bandwidth will be wasted. However, this mis-allocation is largely avoided by always trying to activate links with low depth first. Algorithm 6 checks whether a link e can be active simultaneously

Algorithm 6 AssignDOF(L, e)

```
Step 1  if ($\exists l \in L, e$ is incident to $l$)
          return $\emptyset$;
        endif

        if ($e = (v_i, v_j)$ does not have secondary interference with any link in $L$)
          return $\{(i,j),(j,i)\}$;
        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 $d$ in $G'$ and the corresponding link flow allocation $Flow$;

Step 3  if $f_{\max} < |V_{TR}|$ return $\emptyset$; endif
        $A \leftarrow \{(i,j),(j,i)\}$;
        forall $e' = (x, y) \in E_2$ (where $x$ corresponds to node $v_k$, $y$ corresponds to
        node pair $(v_h, v_l)$)
          if ($Flow[e'] = 1$ and $k = h$)
            $A \leftarrow A \cup \{(h,l)\}$;
          else $A \leftarrow A \cup \{(l,h)\}$;
          endif
        endforall
        return $A$;
```

with the link set L. In Step 1, we check whether there exists primary interference between e and some 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,[6] 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 V_T and

[6]Includes the case when $L = \emptyset$

V_R to denote the set of transmitters and receivers corresponding to link set $L \cup \{e\}$ respectively. We also define a node pair set $V_{TR} = \{(v_i, v_j) | v_i \in V_T, v_j \in V_R, v_i \in N_j, (v_i, v_j) \notin L \cup \{e\}\}$. 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 has 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 $E = E_1 \cup E_2 \cup E_3$. Fig. 4 shows an example of auxiliary graph G'.

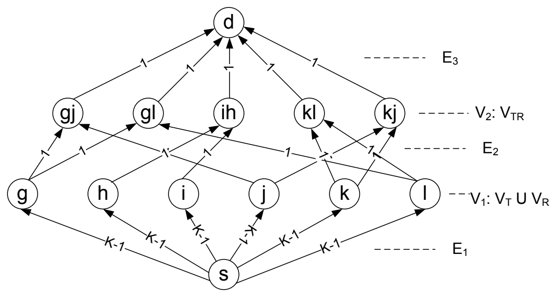


Figure 4. An example of auxiliary graph.

In G', each vertex in E_2 actually correspond to a possible secondary interference. We create two edges for such a vertex because a possible 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 augmenting path based maximum flow algorithm such as 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 $|V_{TR}|$ ($|E_2|$) in G', then there exists a feasible DOF assignment such that all possible secondary interference can be cancelled. Otherwise, we know the given link e cannot be active concurrently with links in L.

We show that a link flow allocation from s to d in G' corresponds to a DOF assignment. Note that each edge $(u, f) \in E_2$ selected in the link flow allocation corresponds to assigning one DOF at node u to eliminate secondary interference f. Next, each edge in E_3 has capacity 1, so each vertex in F can contribute at most 1 to the flow. This corresponds to the fact that a secondary interference only need to be eliminated once, either by transmitter or by receiver. Finally, each edge in E_1 has capacity $N_i - 1$, so each vertex in $V_T \cup V_R$ can contribute at most $N_i - 1$ to the flow. This corresponds to the number of DOFs at each node that can be used to eliminate secondary interferences.[7] In Step 2, we find the maximum flow in G', which corresponds to a DOF assignment to eliminate maximum number of secondary interferences. If this flow number is less than the number of secondary interferences among the links, it is impossible to eliminate all secondary interferences. Otherwise, the algorithm returns a DOF assignment A in Step 3, where each $(x, y) \in A$ means to assign a DOF of node x pointing at node y.

In Algorithm 6, Step 1 takes $O(|L|^2)$ time. In Step 2, G has $2|L| + |F| + 2$ vertices and $2|L| + 3|F|$ edges. The Ford-Fulkerson algorithm in Step 3 takes $O(|E'|f_{\max})$ time. Since in G', $f_{\max} \leq |F|$, Step 3 takes $O((|L| + |F|)|F|)$ time. Step 4 loops for $|f_{\max}|$ times and each loop takes $O(1)$ time. Overall, Algorithm 6 takes $O(|L|^2) + O(|L| + |F|) + O((|L| + |F|)|F|) + O(|F|) = O(|L|^2 + |F|^2)$ time.

In Algorithm 5, Step 1 takes $O(n^2)$ time to initiate $\mathbf{\Gamma}$. Step 2 sorts $e \in E$ in $O(n \log n)$ time, and for each $e \in E$ runs Algorithm 6 once, which takes $O(|L|^2 + |F|^2) = O(n^2)$ time. The Step 2 and 3 will execute for T^u times. Therefore, the

[7]Our algorithm can be applied to networks where nodes have different number of DOFs.

overall time of the algorithm is $O(n^2) + T^u(O(n \log n + nO(n^2)) = O(n^3 T^u)$. However, in practice, the set of links that can be active in a minislot L and the set of secondary interferences F are usually much smaller than $|E|$, the total number of links. Therefore, the algorithm has decent performance.

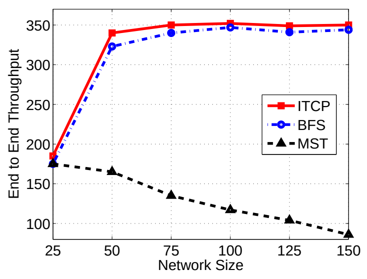
PERFORMANCE EVALUATION

The performance of the proposed algorithms was evaluated via extensive simulations. The simulations were implemented using Microsoft Visual C++ and LEDA, a graph library. In the simulations, the nodes were uniformly deployed within a 4×8 km^2 rectangular region, with a BS at the top-left corner. The number of nodes (network size) varied from 25 to 150, with step size of 25. According to [1], the number of minislots per frame was set to 1024. The uplink and downlink demands of an SS were uniformly distributed in $[5, 10]$ and $[10, 20]$ respectively. The transmission and interference range was set to 1 km and 3 km respectively.

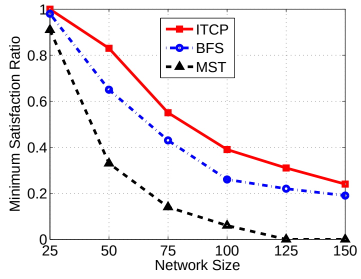
Since our work is the first to address WiMAX scheduling with smart antennas, we compared our scheduling algorithms with the first-fit algorithm and a trivial solution. The first-fit algorithm is a typical greedy algorithm, which tries to pack as many links with unsatisfied bandwidth demands as possible 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 those by MST and BFS. The end-to-end throughput, the minimum satisfaction ratio and the well-known Jain's fairness index [23]1 (in terms of satisfaction ratio) for the uplink traffic are used as the performance metrics.

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. 5. 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 is always used to construct the routing tree. The corresponding results are presented in Figs. 6 and 7

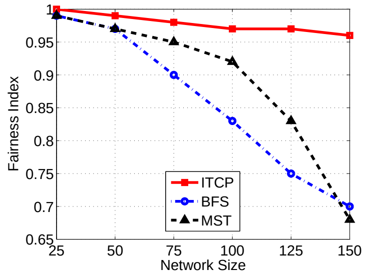
1fairness $= \frac{(\sum_i x_i)^2}{n \sum_i x_i^2}$



(a) End-to-end throughput

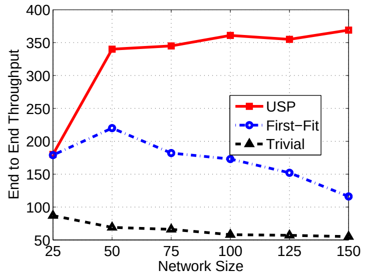


(b) Minimum satisfaction ratio

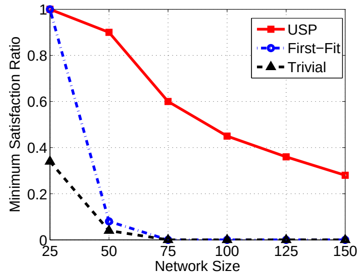


(c) Fairness index

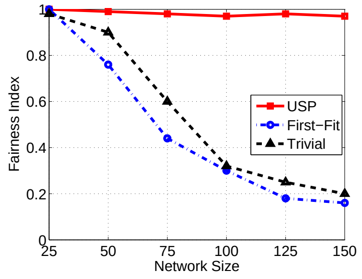
Figure 5. The tree construction algorithms.



(a) End-to-end throughput



(b) Minimum satisfaction ratio



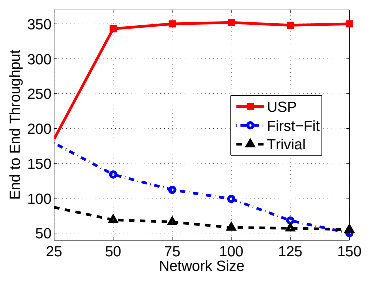
(c) Fairness index

Figure 6. The scheduling algorithms for the special case.

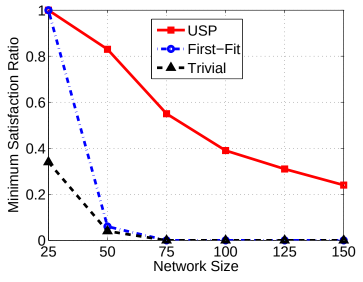
respectively. In scenario 4, we evaluated the performance of different complete solutions (scheduling + routing). Refer to Fig. 8 for the results. For scenarios 1, 3, and 4, the number of DOFs at each node was set to $K = 3$. Each result presented in the figures is the average over 100 simulation runs. In each run, a network is randomly generated and used for all algorithms. In these figures, "USP" stands for our uplink scheduling algorithm for the special and general cases and "ITCP" represents our tree construction algorithm.

We make the following observations from Figs. 5–9:

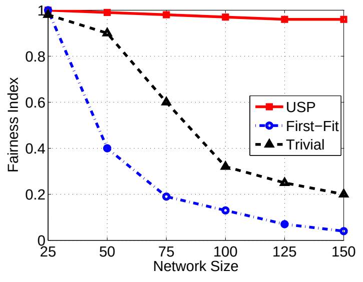
1. As shown in Fig. 5, compared to the BFS and MST algorithms, our tree construction algorithm improves the minimum satisfaction ratio by 21% and 120%, the fairness index by 15% and 10%, and the end-to-end throughput by 3% and 140%, respectively. Essentially, more links in an end-to-end path (larger tree



(a) End-to-end throughput

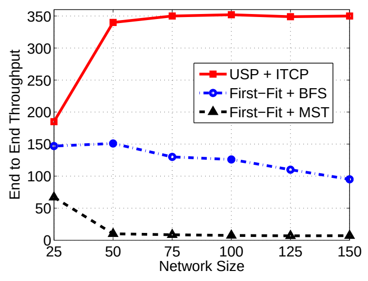


(b) Minimum satisfaction ratio

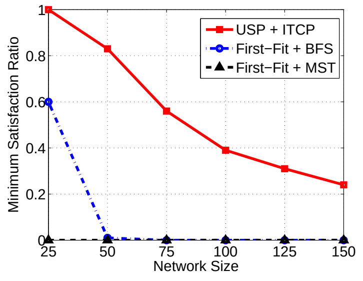


(c) Fairness index

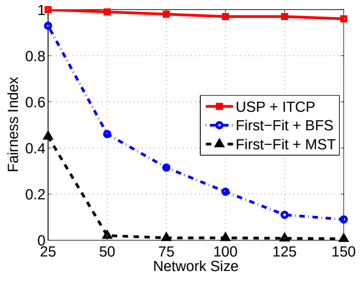
Figure 7. The scheduling algorithms for the general case.



(a) End-to-end throughput

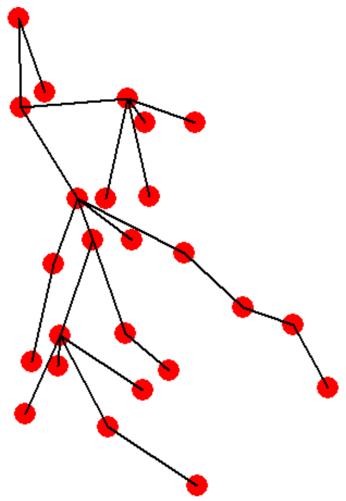


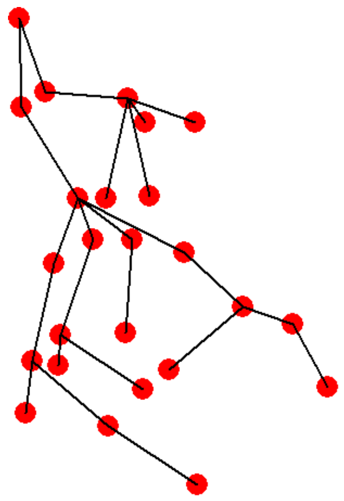
(b) Minimum satisfaction ratio



(c) Fairness index

Figure 8. The complete solutions.


(a) Breadth-First-Search


(b) Our ITCP algorithm

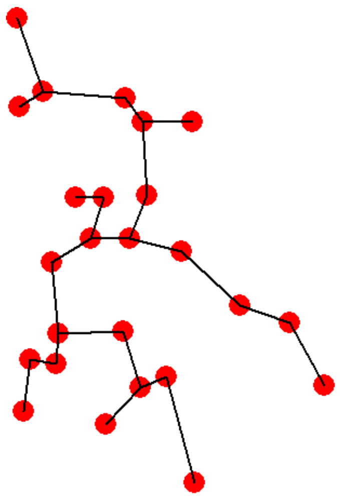
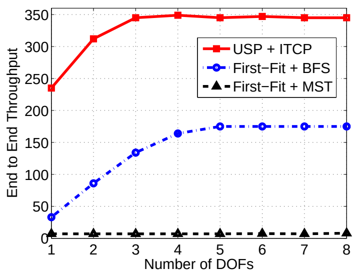

(c) Minimum Spanning Tree

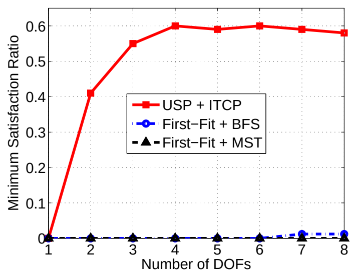
Figure 9. Example trees contructed by different algorithms.

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 tree usually has a larger height than the tree constructed by our routing algorithm. A BFS tree has smaller height than an MST tree. However, the BFS trees are usually imbalanced, i.e., a particular node may have a relatively large number of descendants, which is obviously a negative factor for achieving good performance. Check Fig. 9 for a typical result of trees constructed by different algorithms.

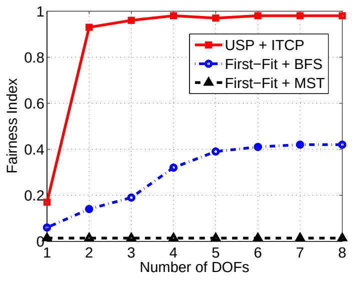
2. Our scheduling algorithms always perform the best in both the special and the general cases. Specifically, compared to the first-fit algorithm, our scheduling algorithm (for the general case) can significantly improve the end-to-end throughput by 213%, the minimum satisfaction ratio by 220%, the fairness index by 200% on average. Moveover, 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. The first-fit algorithm performs closely to our algorithm when the network size is small and almost all requests can be satisfied. As the network size grows, the first-fit algorithm acquires a higher end-to-end throughput, but with a lower fairness index and a very low minimum satisfaction ratio which means some of the nodes get very little or even no bandwidth allocated. As network size keeps growing and the network becomes saturated, the first-fit algorithm even has throughput decreased. This is because in such cases, first-fit allocates most minislots to links at highest levels, while links at lower levels almost do not get any minislot. Therefore, lots of minislots are wasted at high-level links because no traffic can be relayed from low-level nodes. Our algorithm can find a balanced tree with relatively small height. The tree constructed by BFS has fewer levels than MST. So most traffic flows have fewer hops to BS, and therefore BFS has better



(a) End to end throughput



(b) Minimum satisfaction ratio



(c) Fairness index

Figure 10. Complete solution performance with different DOFs.

throughput than MST. However, BFS has the worst fairness index because it is likely for a node to have a very large subtree and result in an imbalanced tree.

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 has heavier traffic demands, and is supposed to result in higher throughput. Therefore, we can see from Figs. 6–8 that the end-to-end throughput given by our scheduling and routing algorithms always increases with the network size. However, more SSs introduce stronger interference, which will hold back the throughput improvement on the contrary. Therefore, the throughput given by algorithms without carefully addressing the impacts of 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 high fairness in a larger network.

In the second scenario, we fix the network size to 75 and examine the performances of three complete solutions with secondary interferences and different number of DOFs from 1 to 8. The results are presented in Fig. 10.

We make the following observations from Fig. 10:

1. All three solutions will have better performance with more DOFs available. However, no matter how many DOFs are available in a node, our complete solution always has the best performance.

2. Because our solution conducts optimal DOF assignment, it can use the fewest DOFs to reach the optimal performance. Therefore, it enables the usage of DAAs with fewer DOFs to acquire interference free and reduces the network infrastructure cost.

CONCLUSIONS

In this project, we studied routing and scheduling in WiMAX 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 that the trees constructed by our algorithm outperform the well-known MST and BFS trees by simulations. 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 first-fit+BFS solution, our interference aware routing and scheduling scheme can improve throughput by 154% and fairness index by 177% on average.

REFERENCES CITED

[1] IEEE 802.16 Work. Group, “IEEE standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems,” *IEEE Std 802.16—2004 (Revision of IEEE Std 802.16—2001)*, pp. 0_1–857, 2004.

[2] K. Sundaresan, W. Wang, and S. Eidenbenz, “Algorithmic aspects of communication in ad-hoc networks with smart antennas,” in *MobiHoc’06: 7th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing*, Florence, Italy, May 22–25 2006, pp. 298–309.

[3] L. Fu, Z. Cao, and P. Fan, “Spatial reuse in IEEE 802.16 based wireless mesh networks,” in *ISCIT’05: IEEE Int. Symp. on Commun. & Inform. Technology*, vol. 2, Beijing, China, Oct. 12–14 2005, pp. 1358–1361.

[4] D. Kim and A. Ganz, “Fair and efficient multihop scheduling algorithm for IEEE 802.16 BWA systems,” in *BroadNets’05: 2nd Int. Conf. on Broadband Networks*, vol. 2, Boston, MA, USA, Oct. 3–7 2005, pp. 833–839.

[5] J. Tao, F. Liu, Z. Zeng, and Z. Lin, “Throughput enhancement in WiMAX mesh networks using concurrent transmission,” in *WCNM’05: IEEE Int. Conf. on Wireless Commun., Networking & Mobile Computing*, vol. 2, Wuhan, Hubei, China, Sept. 23–26 2005, pp. 871–874.

[6] H.-Y. Wei, S. Ganguly, R. Izmailov, and Z. J. Haas, “Interference-aware IEEE 802.16 WiMAX mesh networks,” in *VTC’05-Spring: IEEE 61st Semiannu. Veh. Technology Conf.*, vol. 5, Stockholm, Sweden, May 30–June 1 2005, pp. 3102–3106.

[7] M. Cao, V. Raghunathan, and P. R. Kumar, “A tractable algorithm for fair and efficient uplink scheduling of multi-hop WiMAX mesh networks,” in *WiMesh’06: 2nd IEEE Workshop on Wireless Mesh Networks*, Reston, VA, USA, Sept. 25 2006, pp. 93–100.

[8] H. Shetiya and V. Sharma, “Algorithms for routing and centralized scheduling to provide QoS in IEEE 802.16 mesh networks,” in *WMuNeP’05: 1st ACM Workshop on Wireless Multimedia Networking & Performance Modeling*, Montréal, Quebec, Canada, Oct. 13 2005, pp. 140–149.

[9] C. Cicconetti, I. F. Akyildiz, and L. Lenzini, “Bandwidth balancing in multi-channel IEEE 802.16 wireless mesh networks,” in *InfoCom’07: 26th IEEE*

Int. Conf. on Comput. Commun., Anchorage, AK, USA, May 6–12 2007, pp. 2108–2116.

[10] K. Sundaresan and S. Rangarajan, "On exploiting diversity and spatial reuse in relay-enabled wireless networks," in *MobiHoc'08: 9th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing.* New York, NY, USA: ACM, 2008, pp. 13–22.

[11] S. Ramanathan, "A unified framework and algorithm for channel assignment in wireless networks," *Wireless Networks*, vol. 5, no. 2, pp. 81–94, 1999.

[12] W. Wang, Y. Wang, X.-Y. Li, W.-Z. Song, and O. Frieder, "Efficient interference-aware TDMA link scheduling for static wireless networks," in *MobiCom'06: 12th Annu. Int. Conf. on Mobile Computing & Networking*, Los Angeles, CA, USA, Sept. 24–29 2006, pp. 262–273.

[13] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vaidya, "Using directional antennas for medium access control in ad hoc networks," in *MobiCom'02: 8th Annu. Int. Conf. on Mobile Computing & Networking*, Atlanta, Georgia, USA, Sept. 23–28 2002, pp. 59–70.

[14] T. Korakis, G. Jakllari, and L. Tassiulas, "A MAC protocol for full exploitation of directional antennas in ad-hoc wireless networks," in *MobiHoc'03: 4th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing*, Annapolis, MD, USA, June 1–3 2003, pp. 98–107.

[15] 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," *Mobile Computing, IEEE Trans. on*, vol. 3, no. 4, pp. 350–365, Oct.–Dec. 2004.

[16] B. Mumey, J. Tang, and T. Hahn, "Joint stream control and scheduling in multihop wireless networks with MIMO links," in *ICC'08: IEEE Int. Conf. on Communications*, Beijing, China, May 2008, pp. 2921–2925.

[17] K. Sundaresan and R. Sivakumar, "A unified MAC layer framework for ad-hoc networks with smart antennas," in *MobiHoc'04: 5th ACM Int. Symp. on Mobile Ad Hoc Networking & Computing*, Roppongi Hills, Tokyo, Japan, May 24–26 2004, pp. 244–255.

[18] M. Hu and J. Zhang, "MIMO ad hoc networks with spatial diversity: medium access control, saturation throughput, and optimal hop distance," in *CDC'04: 43rd IEEE Conf. on Decision & Control*, vol. 3, Atlantis, Paradise Island, Bahamas, Dec. 14–17 2004, pp. 3301–3306.

[19] R. Bhatia and L. Li, “Throughput optimization of wireless mesh networks with MIMO links,” in *InfoCom'07: 26th IEEE Int. Conf. on Comput. Commun.*, Anchorage, AK, USA, May 6–12 2007, pp. 2326–2330.

[20] Y.-H. Lin, T. Javidi, R. L. Cruz, and L. B. Milstein, “Distributed link scheduling, power control and routing for multi-hop wireless MIMO networks,” in *ACSSC'06: 40th Asilomar Conf. on Signals, Syst. & Comput.*, Pacific Grove, CA, USA, Oct. 29–Nov. 1 2006, pp. 122–126.

[21] R. R. Choudhury and N. H. Vaidya, “Deafness: a MAC problem in ad hoc networks when using directional antennas,” in *ICNP'04: 12th IEEE Int. Conf. on Network Protocols*, Oct. 2004, pp. 283–292.

[22] K. Jain, J. Padhye, V. N. Padmanabhan, and L. Qiu, “Impact of interference on multi-hop wireless network performance,” *Wireless Networks*, vol. 11, no. 4, pp. 471–487, 2005.

[23] R. Jain, D. M. Chiu, and W. Hawe, “A quantitative measure of fairness and discrimination for resource allocation in shared computer systems,” 1984. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cs/9809099