2016

Link schedule construction algorithms for multi-transmit-receive wireless mesh networks

He Wang
University of Wollongong, hw407@uowmail.edu.au

Link Schedule Construction Algorithms for Multi-Transmit-Receive Wireless Mesh Networks

A thesis submitted in partial fulfillment of the requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

He Wang

Bachelor of Engineering (Telecommunications)

School of Electrical, Computer and Telecommunications Engineering

May 2016

Statement of Originality

I, He Wang, declare that this thesis, submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, is wholly my own work unless otherwise referenced or acknowledged. The document has not been submitted for qualifications at any other academic institutions.

Signed

He Wang

February, 2016

Abstract

The capacity of Wireless Mesh Networks (WMNs) is closely tied to the number of links activated simultaneously. More active links mean wireless routers have a higher data forwarding rate to one or more gateways and also other routers. The key challenge, however, is to limit interference with the goal of maximizing the number of active links or spatial reuse. To this end, researchers have proposed to equip nodes with Multiple-Transmit-Receive (MTR) capability. Briefly, MTR capability can be achieved using the following methods: (i) equipping nodes with multiple directional antennas, (ii) using Multiple-Input-Multiple-Output (MIMO) technology, and (iii) exploiting 60 GHz radios, or the mmWave band. As a result, nodes can work in either i) half-duplex mode, where nodes are able to transmit *or* receive on *all* their links simultaneously, or ii) full-duplex mode, where nodes are able to transmit to *and* receive from multiple neighbors concurrently.

Given an MTR-capable WMN, a key problem is to develop an efficient link scheduler that fully exploits the MTR capability of nodes in order to maximize network capacity. Henceforth, this thesis aims to develop novel Time Division Multiple Access (TDMA) based link schedulers for both half-duplex and full-duplex MTR WMNs. The overarching aim is to generate a short TDMA schedule or superframe consisting of multiple time slots that are repeated periodically. The problem is significant because a short superframe results in links being activated more frequently,

and thus a WMN will have a higher network capacity.

The *first* contribution in this thesis is a distributed approach to derive a minimal superframe. Advantageously, it determines the set of links in each slot using only local information. Compared with centralized approaches, distributed schedulers do not require global information and rely solely on local information to derive a TDMA schedule. This is advantageous because such schedulers allow nodes to adapt to changes quicker, and they scale better with growing number of nodes. To this end, this thesis presents Algo-d, the first distributed scheduler for MTR WMNs that aims to derive a minimal superframe whilst maximizing the number of concurrent links activated in each slot. Specifically, Algo-d divides all nodes into two maximally connected sets, where the number of unscheduled links across these sets are maximized in a distributed manner. Further, Algo-d uses a novel distributed protocol whereby the node with the lowest ID is responsible for notifying other nodes to start the data transmission process.

A key consideration when scheduling links is traffic flow. In particular, traffic flowing to/from a gateway; i.e., uplink and downlink packets. It is important that a gateway delivers downlink packets to their respective destination and nodes forward any buffered packets to the gateway in minimum time. To date, existing schedulers that consider traffic to/from a gateway do not consider the benefits brought by MTR. Henceforth, the *second* contribution in this thesis is a scheduler called Algo-PB that derives a minimal schedule to deliver packets from a gateway to destination nodes in MIMO-based MTR WMNs. Unlike past works, Algo-PB takes advantage of MTR and spatial multiplexing to generate a schedule that forwards packets to/from gateways quickly. This thesis also extends Algo-PB to consider multiple gateways. This extension, called the forest construction problem, aims to build a tree rooted at each gateway such that the makespan of the longest personalized broadcast schedule is minimized. An Integer Linear Program (ILP) is proposed to determine the optimal routing forest that minimizes the longest personalized broadcast schedule. As the complexity of the ILP increases exponentially with network size, a heuristic algo-

rithm called Algo-FC is then proposed to generate a balanced forest by repeatedly picking the tree with the heaviest load and migrating a node from it to another tree if doing so reduces its load.

The last contribution relates to advances in full-duplex communications over the same frequency and also the observation that both uplink and downlink packets can be scheduled jointly if nodes can transmit *and* receive packets simultaneously. Henceforth, this thesis considers WMNs with MIMO full-duplex nodes. In particular, these nodes assign one or more antenna elements to cancel self and neighboring interference in order to maximize the number of transmissions/receptions. Two TDMA based link schedulers are presented: UDMAC and Algo-UD. UDMAC is a path-by-path method. It generates a link schedule by repeatedly scheduling the packet that is the farthest from its destination. Algo-UD, on the other hand, generates a schedule in a slot-by-slot manner. In each slot, Algo-UD schedules as many packets as possible for each node, starting from the node with the most number of packets. Moreover, Algo-UD uses a novel node ordering rule to determine nodes responsible for suppressing interference.

Acknowledgments

I like to take this opportunity to thank people who guided and supported me during my research. Without their contributions, this research would never be possible.

First and foremost, I would like to express my special appreciation to my supervisor, Associate Professor Kwan-Wu Chin, a tremendous mentor for me. I would like to thank for his patience, selfless guidance, constructive suggestions, constant encouragement and high requirements, which have motivated me to grow into an independent researcher.

I like to thank my co-supervisor: Dr Raad Raad and Dr Sieteng Soh, for their contributions and helpful comments in the past three and half years.

I also like to thank the reviewers and editors of my conference and journal papers, for their constructive comments, which have improved the presentation and quality of my publications significantly.

Special thanks go to my friends and research mates in my office, Dr. Luyao Wang, Dr. Changlin Yang, Mr Shichao Fu, Mr Sen Zhang, Mr Tengjiao He, Mr. Tianle Liu and Miss Heidi Xu for the joyful conversations and pleasant working environment.

Lastly, I would like to thank my family members. Words cannot express how grateful I am to my parents for all their love and support. Without their love, I would not have reached this milestone.

Contents

Abstract II

Acknowledgments V

Abbreviations XIII

1 Introduction 1

1.1 Problem Space 6

1.1.1 Distributed Link Scheduling 6

1.1.2 Personalized Broadcast Scheduling 7

1.1.3 Joint uplink and downlink packet scheduling 10

1.2 Contributions 11

1.2.1 Distributed Maximal Link Scheduler 12

1.2.2 Personalized Broadcast Scheduler 12

1.2.3 Joint Uplink and Downlink Packet Scheduler 13

1.3 Publications 14

1.4 Thesis Structure 14

2 Literature Review 16

2.1 TDMA WMNs with Directional Antennas 16

2.1.1 Single-Radio 16

2.1.2 Multiple Radios . . . 28
2.1.3 Smart Antennas . . . 34
2.2 Personalized Broadcast and Data Collection Scheduling . . . 40
2.3 Forest Construction . . . 45
2.4 Full-Duplex . . . 50
2.5 Summary . . . 58

3 A Distributed Maximal Link Scheduler for Multi Tx/Rx Wireless Mesh Networks 60

3.1 Preliminaries . . . 61
3.2 The Problem . . . 63
3.3 Solution . . . 64
3.3.1 Channel Access . . . 65
3.3.2 Link Scheduling . . . 66
3.3.3 Opportunistic Links . . . 71
3.3.4 Superframe Start Time . . . 72
3.4 Analysis . . . 74
3.5 Evaluation . . . 78
3.5.1 Average Collision Per Node . . . 80
3.5.2 Schedule Construction Cost . . . 81
3.5.3 Node Density . . . 82
3.5.4 Node Degree . . . 87
3.6 Conclusion . . . 87

4 On Personalized Broadcast and Forest Construction in Multi Tx/Rx Wireless Mesh Networks 89

4.1 Preliminaries . . . 90
4.2 Upper and Lower Bound . . . 92
4.3 Personalized Broadcast Scheduling . . . 97
4.4 Forest Construction . . . 102

4.4.1 An ILP . . . 102
4.4.2 Heuristic Solution . . . 104
4.5 Analysis . . . 108
4.6 Evaluation . . . 110
4.6.1 Algo-PB . . . 110
4.6.1.1 Node Demand . . . 111
4.6.1.2 Node Density . . . 111
4.6.1.3 Number of Antennas . . . 112
4.6.2 Forest Construction . . . 113
4.6.2.1 Network Size . . . 115
4.6.2.2 Node Density . . . 115
4.6.2.3 Number of Gateways . . . 118
4.7 Conclusion . . . 118

5 On Uplink and Downlink Packet Scheduling in Full-Duplex Wireless Mesh Networks 120

5.1 Preliminaries . . . 121
5.2 Upper and Lower Bound . . . 123
5.3 Solution . . . 125
5.3.1 UDMAC . . . 125
5.3.2 Algo-UD . . . 129
5.4 Evaluation . . . 133
5.4.1 UDMAC . . . 134
5.4.1.1 Node Density . . . 134
5.4.1.2 Node Demand . . . 134
5.4.1.3 Number of Antennas . . . 136
5.4.2 Algo-UD . . . 136
5.4.2.1 Node Demand . . . 137
5.4.2.2 Node Density . . . 139

5.4.2.3 Number of Antennas 139
5.5 Conclusion 140

6 Conclusion **142**

References **145**

Appendices **158**

List of Figures

1.1 An example WMN and its corresponding link schedule 2
1.2 An example WMN 6
1.3 An example of data forwarding in WMNs 7
1.4 An example of data forwarding in WMNs 8
1.5 Personalized broadcast schedule for a WMN with (a) a single tree or (b) forest 9
1.6 An example showing NI 10
1.7 An example Joint-Schedule with full-duplex nodes. 11

2.1 An example WMN and its corresponding routing forest 46
2.2 An example of relay and bi-directional full-duplex transmissions . . . 50

3.1 An example of the *no-Tx-Rx* constraint 62
3.2 An example WMN topology and its corresponding schedule 63
3.3 State diagram of Algo-d 67
3.4 Message exchange in the first slot 70
3.5 Two-boxes topology 71
3.6 Improving a schedule with opportunistic links. Solid arrows are scheduled links and dotted arrows are opportunistic links 73

3.7 Average number of collisions experienced by a node under various node density. . . . 80
3.8 Average number of collisions experienced by a node under various node degrees. . . . 81
3.9 Average number of message exchanged per node to generate a schedule under various node degrees. . . . 82
3.10 Time to generate a schedule under various node degrees. . . . 82
3.11 Performance under different node densities . . . 84
3.12 Performance under different node degrees . . . 86

4.1 Example topology . . . 95
4.2 Example topology . . . 101
4.3 Example topology . . . 107
4.4 Schedule length under different node demand . . . 111
4.5 Schedule lengths under different network sizes . . . 112
4.6 Schedule length versus Δ_u . . . 113
4.7 Performance under different network sizes . . . 116
4.8 Performance under different node densities . . . 117
4.9 Performance under different number of gateways . . . 119

5.1 A tree topology with interfering link (dotted lines). . . . 124
5.2 Performance evaluation under different node density. . . . 135
5.3 Performance evaluation under different node demands. . . . 135
5.4 Performance evaluation under different number of antennas. . . . 136
5.5 Schedule length versus number of packets per node . . . 138
5.6 Schedule length under different node density . . . 140
5.7 Schedule length versus number of antennas per node . . . 141

List of Tables

1.1 A superframe with half-duplex nodes 6
1.2 A superframe with full-duplex nodes 6
1.3 TDMA schedule for Figure 1.7 11

2.1 Comparison of works where nodes have a single directional antenna . 27
2.2 Comparison of works using multi-radio directional antennas 35
2.3 Comparison of works using smart antennas 41
2.4 Comparison of recent personalized broadcast and data collection schedulers 45
2.5 Comparison of recent forest construction and load balance routing schemes 49
2.6 Comparison of recent full-duplex MAC protocols 57

3.1 Notations and definitions 62
3.2 Description of messages 66
3.3 Description of each state 66
3.4 The (x, y) value of each node in the first slot 70

4.1 Link schedule for Figure 4.2 101

5.1 Schedule for Figure 1.7 133

Abbreviations

AAC	Active Antenna Cancellation
ACK	Acknowledgment
ADC	Analog to Digital Converter
AP	Access Point
BDA	Bucket Draining Algorithm
BFS	Breadth First Search
BIP	Binary Integer Programming
BS	Base Station
CSI	Channel State Information
CSMA	Carrier Sense Multiple Access
CTS	Clear to Send
DEC	Directed Edge Coloring
DoF	Degree of Freedom
FIFO	First In First Out
GPS	Global Positioning System
HWF	Heavy Weighted First
IC	Interference Cancellation
ILP	Integer Linear Programming
LP	Linear Programming

MAC	Medium Access Control
MBAA	Multi-beam Adaptive Array
MDF	Max Degree First
MILP	Mixed Integer Linear Programming
MIMO	Multiple-Input-Multiple-Output
MTR	Multiple-Transmit-Receive
MU-MIMO	Multi-user MIMO
MUI	Multi-User Interference
NI	Neighboring Interference
NIC	Network Interface Card
PAA	Phased Antenna Array
PNC	Piconet Controller
QoS	Quality of Services
RTS	Request to Send
RSSI	Received Signal Strength Indicator
SI	Self-Interference
SINR	Signal to Interference and Noise Ratio
TDMA	Time Division Multiple Access
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network
WSN	Wireless Sensor Network

Chapter 1

Introduction

Wireless Mesh Networks (WMNs) have developed rapidly in recent years to provide 'last miles' Internet access in both urban and rural areas [1]. A WMN is comprised of an ad hoc collection of nodes with one or more radios [2]. These nodes operate not only as a host, but also as a router, whereby they collaboratively send and forward packets to a gateway or other hosts [3]. Figure 1.1 shows an example WMN. Solid circles indicate mesh nodes and squares are gateways. A key feature of WMNs is that nodes help each other forward packets over multiple hops. For example, we see that packets from node A being forwarded over two hops to gateway G. A key advantage of WMNs is that mesh routers can be deployed in an ad-hoc manner to extend coverage or to improve reliability. An example is using IEEE 802.11s [4] to extend the coverage of existing Wireless Local Area Networks (WLANs). Advantageously, mesh routers can self-organize and self-configure after deployment to establish one or more routes to each other [3]. Critically, they may configure a transmission schedule that ensures links are activated in a collision-free manner, and packets are transmitted to/from a gateway quickly.

The link schedule used by a WMN has a direct relationship with the number of links that are activated concurrently; equivalently, the capacity of a WMN. More active links over a given time interval means a WMN is forwarding more data. To

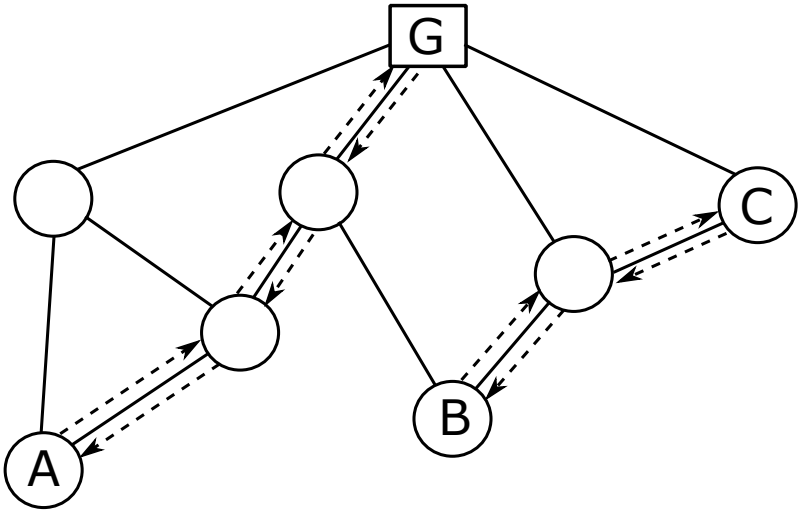


Figure 1.1: An example WMN and its corresponding link schedule

date, researchers have proposed a number of approaches to improve the capacity of a WMN; see [1] and references therein for a survey. This thesis is no different. It considers a new type of WMN where mesh routers have Multiple-Transmit-Receive (MTR) capability; i.e., mesh routers are either (i) half-duplex, where they transmit *or* receive on all their links, or (ii) full-duplex, where they transmit *and* receive to/from their neighbors concurrently. In both cases, links operate in the same frequency.

To date, three methods can be used to endow nodes with MTR capability. The first method is to equip mesh routers with multiple directional antennas. Nodes with directional antennas, unlike those that use an omni-directional antenna, are able to focus their transmission beam towards a specific direction. These antennas help reduce interference between nearby transmissions and allow multiple pairs of nearby nodes to communicate simultaneously if their beams do not overlap. This in turn increases network capacity due to higher spatial reuse [5]. To this end, the authors of [6] equip nodes with multiple off-the-shelf IEEE 802.11 radios and parabolic antennas. To enable MTR, they separate adjacent antennas on nodes by at least 30 degrees adjust transmission power to reduce side-lobe interference. They also disable the carrier sense on each Network Interface Card (NIC) so that a node is able to transmit concurrently. Lastly, each NIC is tuned to a common frequency

and has a transmit power level that ensures the signal to interference and noise ratio (SINR) at a receiving node is above a given threshold; note, the transmit power over each link is fixed after deployment.

The second method uses Multiple-Input-Multiple-Output (MIMO) technology [7]. Nodes are equipped with an adaptive antenna array. The signal transmitted/received at each antenna element is split/combined to achieve higher capacity and/or reliability [8]. In general, a MIMO system has the following characteristics: array gain, spatial diversity, spatial multiplexing and interference reduction [9]. Specifically, array gain is achieved by adjusting the main lobe of the receiving beam such that it has high directivity and gain. With spatial diversity, the same signal is transmitted through different antenna elements or Degree of Freedoms (DoFs) to take advantage of independent fading paths. The receiver then combines the arriving signals to increase transmission quality. Another key characteristic of MIMO is that different antenna elements can be used to carry independent data streams. As a result, the capacity of a MIMO system increases linearly with the number of antenna elements. Lastly, a node is able to use its antenna elements to null/suppress co-channel interference caused by neighboring transmitters. Consequently, as demonstrated in [10], Multi-user MIMO (MU-MIMO) has high spatial re-use as well as allowing the transmission/reception of multiple independent data streams to/from different neighbors simultaneously.

The third method exploits the 60 GHz or mmWave band [11]. An important feature of the 60 GHz band is its high propagation loss. Indeed, links in 60 GHz systems have high directivity and can be considered as pseudo-wires [12]. Consequently, the interference caused by neighboring transmissions can be ignored [13]. This results in very high spatial reuse. In addition, higher transmission frequencies lead to very high link capacity and small antennas [14]; these characteristics make the 60 GHz band attractive for use in future 5G cellular systems. An example MTR WMN that uses 60 GHz radios can be found in [15]. The authors equip each node with a phased antenna array (PAA). Each antenna element is equipped with a phase

shifter to adjust the radiating beam. A node can thus beamform to multiple distinct neighbors concurrently. Advantageously, as the 60 GHz band has high path loss, there is little interference between these neighbors.

The three methods thus far assume nodes either transmit *or* receive at a given time. This is because the transmission from a node causes severe self-interference (SI) that prevents it from receiving any incoming signal correctly. This changed in [16] and [17] where the authors show, via a prototype, full-duplex nodes that are capable of transmitting *and* receiving from one or more neighbors simultaneously over the *same* frequency. Another full-duplex system is outlined in [18]. The authors achieve full-duplex communications via active antenna cancellation (AAC). In particular, the authors use a MU-MIMO system. For each receiving antenna, they use one extra transmitting antenna to null any self-interference at a receiving antenna. Besides [18], other works have reported prototypes that suppress SI by at least 95 dB; see [19] [20] and [21]. This is critical to ensure a node is able to decode the much weaker incoming signal. For example, in [20], the authors use both analog and digital cancellation schemes to achieve 110 dB SI reduction. In general, SI cancellation can be carried out passively or actively [22]. Passive cancellation schemes attenuate the transmitted signal or SI. In [16], the transmitted signal is split between two transmit antennas. The authors placed a receive antenna between two transmit antennas. The distance from the two transmit antennas to the receive antenna differs by an odd multiple of half wavelength. This causes the signal transmitted by the two transmitting antennas to add destructively and cancels one another and provides around 30 dB of signal cancellation. Active SI cancellation schemes include analog and digital cancellation [22]. Their first step is analog cancellation where SI is suppressed before the received signal enters the analog to digital converter (ADC). For example, in [17], the authors first create a replica of the transmitted signal using a Balun transformer. They then combine the delayed and attenuated replica signal together with the received signal to reduce SI. The second step is digital cancellation. It aims to suppress any residual SI in the digital domain after the signal passes

the ADC. In [17], the authors model the wireless channel and analog cancellation circuit as a single SI channel. They then use known training symbols at the start of a transmitted packet to estimate the response of the SI channel. Nodes suppress residual SI by subtracting the estimated SI signal in the digital domain.

To take advantage of the advances in MTR technologies, nodes require an appropriate link scheduler. In this regard, one can use Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). However, as shown in [23], a Time Division Multiple Access (TDMA) schedule increases the throughput of a WMN by three to 11 fold. Moreover, an optimal TDMA schedule serves as the asymptotic bound for CSMA-based methods.

Henceforth, this thesis focuses on TDMA based link schedulers that aim to derive a schedule or superframe that has the fewest number of slots. The derived superframe repeats periodically. Each link is assigned one or more transmission slots. Note that links activated in each slot are conflict-free. Moreover, they operate over the same frequency. A link scheduler must also consider the communication capability, i.e., half or full duplex, of nodes. Consider Figure 1.2. Suppose each node A, B and C has one packet to be delivered to gateway G. Table 1.1 shows an example superframe if all nodes can only transmit *or* receive on their links. The superframe consists of three time slots. The link between node B and A and the link between node A and G are scheduled into different time slots. This is because node A cannot transmit *and* receive concurrently. However, if nodes are able to transmit *and* receive on all their links simultaneously, i.e., have full-duplex capability, the superframe is reduced to two slots, see Table 1.2. The link between B and A and the link between A and G can be scheduled in one slot because node A is able to transmit *and* receive simultaneously.

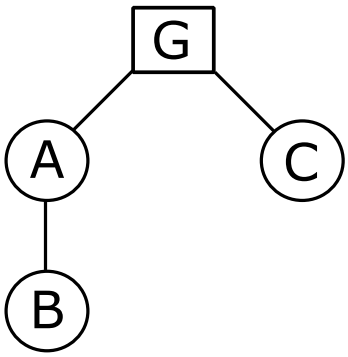


Figure 1.2: An example WMN

Slot 1	**Slot 2**	**Slot 3**
$A \rightarrow G$	$B \rightarrow A$	$A \rightarrow G$
$C \rightarrow G$		

Table 1.1: A superframe with half-duplex nodes

Slot 1	**Slot 2**
$A \rightarrow G$	$A \rightarrow G$
$C \rightarrow G$	
$B \rightarrow A$	

Table 1.2: A superframe with full-duplex nodes

1.1 Problem Space

This thesis focuses on WMNs where nodes have MTR capability. Specifically, each node is able to transmit to and/or receive from multiple neighbors simultaneously. Given an MTR WMN, an efficient link scheduler is required to exploit the benefits brought by MTR and thus maximizes network capacity. In this context, there are a number of fundamental problems to be addressed.

1.1.1 Distributed Link Scheduling

A superframe can be generated centrally at a gateway node or in a distributed manner. In both methods, the goal is to derive a superframe that consists of the minimal number of slots whilst maximizing the number of links activated in each time slot. As will be discussed in Chapter 2, to date, there exist several centralized link scheduling approaches. However, the computation cost of centralized solutions

increases exponentially with network size, meaning they are impractical in large scale WMNs. This thesis thus seeks a distributed solution. Compared with centralized schedulers, distributed schedulers do not require global topological information. This means nodes are able to work with local, e.g., 2-hops, information to derive a superframe or TDMA schedule. This is advantageous because it requires orders of magnitude less computation and ensures a link scheduler scales with increasing nodes [24] [25]. Apart from that, by virtue of being distributed, no central server is required and adds to the self-organizing and self-configuring features of WMNs [3].

1.1.2 Personalized Broadcast Scheduling

In a WMN, data forwarding to/from one or more gateways is a fundamental operation. In practice, the gateway(s) and mesh routers will have packets to deliver. To this end, it is critical that a gateway delivers buffered packets to their respective receiver or mesh router, aka downlink packets, quickly. Conversely, it is important routers send buffered packets, aka uplink packets, to a gateway promptly.

A data forwarding example is shown in Figure 1.3; assume each node is allowed to transmit *or* receive only one packet at a time. The gateways are denoted as s_1 and s_2. Dashed lines denote packet transmissions and nodes C, D and E are receivers or mesh routers. In this example, two slots are sufficient for the gateways to deliver one packet each to node C, D and E. Note that a short schedule is important to allow the gateways to drain buffered packets quickly.

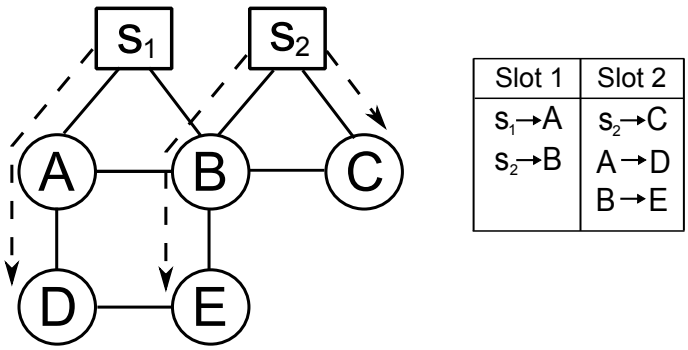


Figure 1.3: An example of data forwarding in WMNs

This thesis aims to derive such a schedule by solving the personalized broadcast problem [26]. Briefly, given an arbitrary tree rooted at a gateway, and each node has a set of packets buffered at the gateway, the aim is to derive a collision-free link schedule to transfer these packets to their receiver in the shortest possible time; i.e., the resulting TDMA schedule/superframe uses the fewest number of slots or makespan. Note, 'personalized broadcast' refers to the fact that the packets to be delivered are unique and has a designated destination. This fact makes it distinct from the broadcast scheduling problem [27] where all nodes receive the *same* packet from a gateway.

Consider Figure 1.4. Nodes can transmit *or* receive one packet at a time. Each non-gateway node, namely A, B, C and D, is awaiting one packet from gateway s. We see that a schedule with four slots is sufficient for non-gateway nodes to receive their respective packet. As an aside, the authors of [26] have shown that the personalized broadcast problem is equivalent to the data collection or data gathering problem. In particular, one can 'reverse' any schedule derived for the personalized broadcast problem and use it for data collection/gathering. That is, links activated in the first, second and subsequent slots in the personalized broadcast schedule are activated last, second last and so forth.

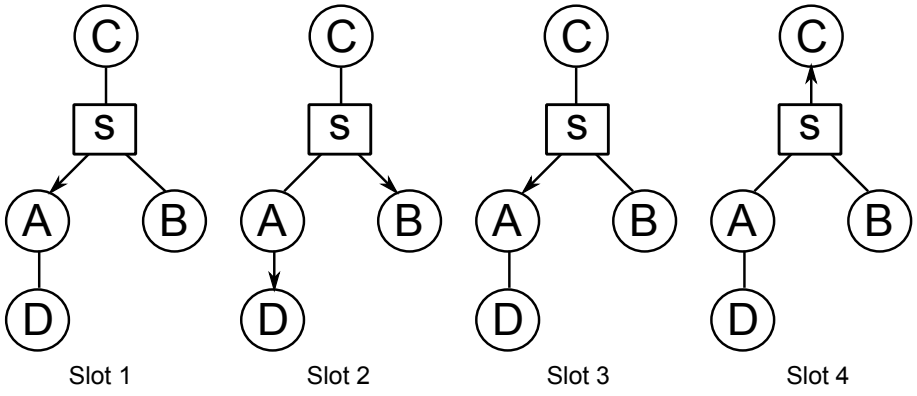


Figure 1.4: An example of data forwarding in WMNs

The previous example only considers one gateway. However, intuitively, one can surmise that multiple gateways may reduce the schedule makespan further. Consider Figure 1.3; recall that s_1 and s_2 are gateways, and each node can transmit or receive

one packet at a time. Assume each non-gateway node has one packet buffered at one of the gateways. If only one gateway, e.g., s_1, is used to transfer packets to non-gateway nodes, the makespan of the personalized broadcast is five slots; see Figure 1.5(a). However, using both s_1 and s_2, the makespan reduces to three slots, shown in Figure 1.5(b). Also, the time to deliver all packets is determined by the personalized broadcast schedule with the longest makespan. This fact thus leads to the forest construction problem, where a forest consists of several trees, each rooted at one gateway. Briefly, for a network with multiple gateways, the aim is to build a forest that minimizes the makespan of the longest personalized broadcast schedule.

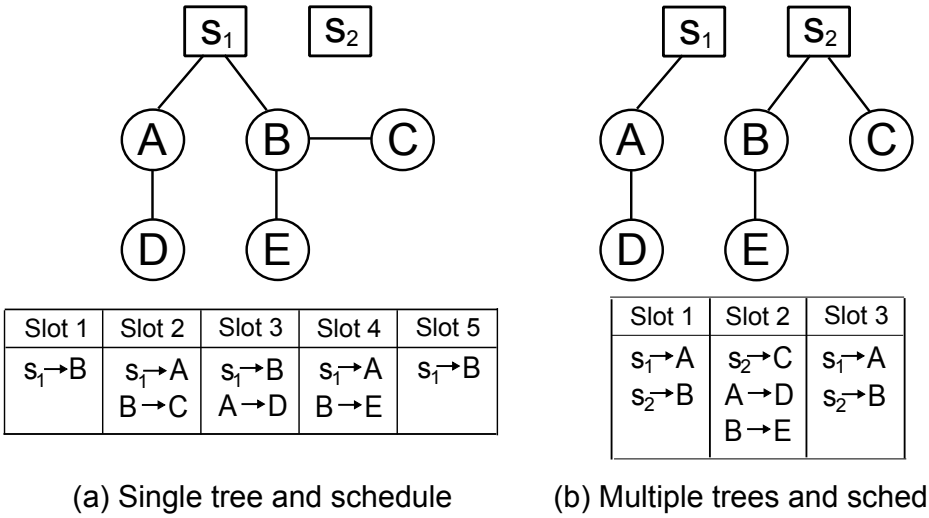


Figure 1.5: Personalized broadcast schedule for a WMN with (a) a single tree or (b) forest

The examples illustrated by Figure 1.4 and 1.5 consider the ideal case where interference is only caused by concurrent transmissions and receptions at one node. However, a transmitting node may interfere with receiving nodes nearby; so called *Neighboring-Interference* (NI). One characteristic of MIMO communications is that a node is able to use its antenna elements to cancel NI to increase spatial reuse. To this end, this thesis adopts the approach of [28] whereby the transmitter or the receiver is responsible for removing SI. Consider Figure 1.6. Solid arrows indicate packet transmissions and dotted ones indicate interference. Suppose node B is within node A's interference range. As node A is transmitting, it causes NI at node B. Assume

node A is transmitting m data streams to its neighbors and B is receiving n data streams from its neighbors. As per [28], if node A is responsible for removing NI it causes to node B, then A needs to use n of its antenna elements to suppress the NI. Further, if node B is required to suppress the NI caused by node A, node B needs to assign m antenna elements to cancel NI.

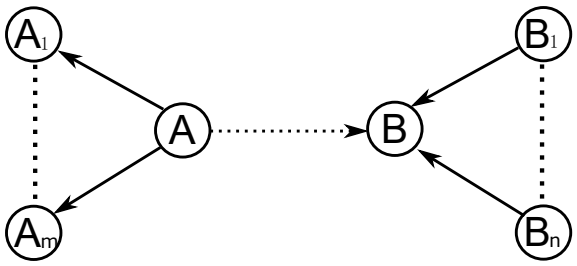


Figure 1.6: An example showing NI

1.1.3 Joint uplink and downlink packet scheduling

To date, past works that address the personalized broadcast or data collection problem assume half-duplex wireless networks. However, with full-duplex capability, nodes are able to transmit *and* receive simultaneously over the same frequency. This means both uplink and downlink packets can be scheduled jointly. To this end, this thesis aims to derive the shortest link schedule, also called Joint-Schedule, that delivers all uplink and downlink packets in full-duplex MU-MIMO based WMNs. Specifically, the aim is to derive a collision-free transmission schedule that enables all nodes, including the gateway, to forward buffered packets to their respective destination using the smallest number of slots.

Consider Figure 1.7. Node 4 is the gateway node. Assume node 1, 2 and 3 have one uplink and one downlink packet, and all nodes have sufficient DoFs for transmission, reception and interference cancellation (IC). Solid arrows indicate packet transmissions and dashed lines indicate NI links. We see that all packets arrive at their destination in two slots.

The previous example assumes each node has sufficient DoFs to transmit, receive and cancel interference. In practice, nodes are likely to have limited DoFs. Conse-

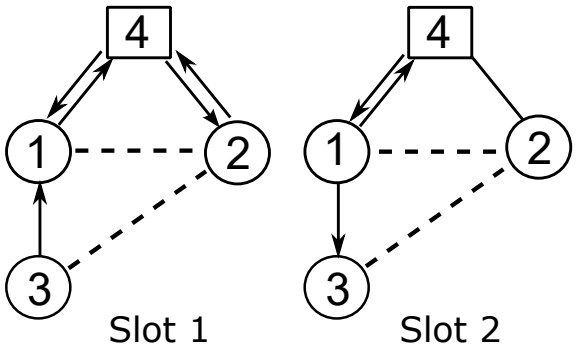


Figure 1.7: An example Joint-Schedule with full-duplex nodes.

	Slot 1	**Slot 2**	**Slot 3**	**Slot 4**
Node 3 (Uplink)	$3 \rightarrow 1$	$1 \rightarrow 4$		
Node 1 (Uplink)	$1 \rightarrow 4$			
Node 2 (Uplink)	$2 \rightarrow 4$			
Node 3 (Downlink)		$4 \rightarrow 1$	$1 \rightarrow 3$	
Node 1 (Downlink)			$4 \rightarrow 1$	
Node 2 (Downlink)				$4 \rightarrow 2$

Table 1.3: TDMA schedule for Figure 1.7

quently, they rely on an efficient rule to assign DoFs during transmissions/receptions. Consider the topology shown in Figure 1.7. Suppose each node 1, 2 and 3 has one uplink and one downlink packet, and all nodes have three DoFs. If uplink packets are scheduled first followed by downlink packets from the node farthest from the gateway with NI suppressed at the transmitter side, the generated schedule is shown in Table 1.3. Note that link $4 \rightarrow 2$ cannot be scheduled in slot 3 as node 1 does not have sufficient DoFs to cancel NI it causes to node 2. Further, if node 2 is responsible for suppressing the NI caused by node 1, link $4 \rightarrow 2$ can then be scheduled in slot 3 and thus, reduces the schedule length by one slot.

1.2 Contributions

This thesis addresses the aforementioned problems and contains a number of solutions. They are summarized as follows.

1.2.1 Distributed Maximal Link Scheduler

To solve the distributed link scheduling problem, this thesis presents Algo-d, an algorithm that aims to derive a minimal superframe by solving the well-known MAX-CUT problem [29] in a slot-by-slot manner. Its goal is to maximize the number of links activated in each time slot. In addition, Algo-d activates each link at least once and maximizes the number of links activated in each slot. In contrast to previous works such as [30] and [31], Algo-d solves the MAX-CUT problem in a distributed manner using only local information. Further, Algo-d uses a novel distributed protocol to determine the node with the lowest ID that is responsible for starting the link scheduling process. Experimental results show that Algo-d provides significantly better performance than distributed algorithms such as JazzyMAC [25] and ROMA [24]. Algo-d shortens the superframe length by 37.5% and increases the number of activated links in each time slot by 270% as compared to JazzyMAC. Also, Algo-d schedules 28% more links as compared to ROMA.

1.2.2 Personalized Broadcast Scheduler

This thesis addresses the aforementioned personalized broadcast problem in MU-MIMO-based WMNs. Critically, there is no previous work that considers nodes with the ability to transmit to or receive from multiple neighbors simultaneously. Moreover, nodes have the option of upgrading a link, by way of multiple antennas, so that more than one packet can be transmitted to a neighbor. In addition, this thesis considers the personalized broadcast problem for MTR WMNs with one or multiple gateways.

This thesis presents Algo-PB, the first link scheduler that generates the minimal personalized broadcast schedule for arbitrary tree topologies constructed in a MTR WMN. Algo-PB generates the personalized broadcast schedule in a path-by-path manner and ensures packets are delivered quickly to nodes; i.e., the resulting throughput is the highest possible. Also, this thesis derives the theoretical person-

alized broadcast makespan lower bound for arbitrary tree topologies.

This thesis also addresses the forest construction problem and models it as an Integer Linear Programming (ILP). It uses the formulation to derive the optimal routing forest. As the complexity of the ILP increases exponentially with network size, this thesis proposes a heuristic, called Algo-FC, to construct the routing forest. Compared with the ILP, Algo-FC has lower complexity and generates a near-optimal balanced forest. Experimental results show that when using Algo-FC, the heaviest gateway has a load that is at most 9.1% more than the optimal solution generated by the ILP. In terms of schedule length, Algo-PB generates up to 45.5% shorter schedule lengths as compared to the state-of-the-art algorithm proposed in [26]. The difference between Algo-PB and the theoretical lower bound is at most 34.5%.

1.2.3 Joint Uplink and Downlink Packet Scheduler

As mentioned, uplink packets and downlink packets can be transmitted concurrently in full-duplex MTR WMNs. To this end, this thesis provides two novel link schedulers called UDMAC and Algo-UD to generate a minimal Joint-Schedule. Specifically, UDMAC is a path-by-path method. Its key idea is to always schedule one packet to/from the node farthest from the gateway that has unscheduled packets. In addition, transmitters are responsible for suppressing NI. Algo-UD is different. It generates the schedule in a slot-by-slot manner. In each slot, Algo-UD schedules as many packets as possible for each node, starting with the node with the highest number of packets. Algo-UD uses the novel node ordering method of [32] to determine whether NI is cancelled by the transmitter or receiver. Also, this thesis presents the theoretical lower and upper bound of the Joint-Schedule for general tree topologies. Experimental results show that UDMAC outperforms the state-of-the-art half-duplex scheduler [26] by at least 60% in terms of schedule length. In addition, the gap between UDMAC and the theoretical lower bound is at most 41.4%. In case of Algo-UD, the schedule generated by Algo-UD is at most 25.7%

shorter than UDMAC.

1.3 Publications

The work in this thesis has resulted in following papers:

1. **H. Wang**, K-W Chin, S. Soh, and R. Raad. *A Distributed Maximal Link Scheduler for Multi Tx-Rx Wireless Mesh Networks*, IEEE International Conference on Communications (ICC), Sydney, Australia, June, 2014.

2. **H. Wang**, K-W Chin and S. Soh. *A Novel Link Scheduler for Personalized Broadcast in Multi Tx/Rx Wireless Mesh Networks*, IEEE 2nd International Workshop on Ad Hoc Networking with MIMO and Cognitive Radio, London, United Kingdom, June, 2015.

3. **H. Wang**, K-W Chin, S. Soh and R. Raad. *A Distributed Maximal Link Scheduler for Multi Tx/Rx Wireless Mesh Networks*, IEEE Transactions on Wireless Communications, 14(1), pp1810-1813, January, 2015.

4. **H. Wang**, K-W Chin, and S. Soh. *On Scheduling Personalized Broadcast in Full Duplex Wireless Mesh Networks*, IEEE Communications Letters, 19(10), pp1810–1813, October, 2015.

5. **H. Wang**, K-W Chin and S. Soh. *On Minimizing Data Forwarding Schedule in Multi Tx/Rx Wireless Mesh Networks*, IEEE Access, vol. 4, pp1570-1582, April, 2016

1.4 Thesis Structure

1. *Chapter 2.* This chapter contains a survey of existing works on centralized and distributed TDMA scheduling algorithms. This chapter also reviews previous works on the personalized broadcast problem and load balance routing.

2. *Chapter 3.* This chapter presents a distributed link scheduler that generates a superframe with minimal length while maximizing the number of concurrent links in MTR WMNs.

3. *Chapter 4.* This chapter presents a novel link scheduler that addresses the personalized broadcast problem and derives a minimal schedule that allows packets to be transmitted to and from a gateway. It also presents the forest construction problem, presents an ILP, and proposes a heuristic algorithm.

4. *Chapter 5.* This chapter considers scheduling uplink and downlink packets simultaneously in full-duplex MTR WMNs. It presents two centralized schedulers that generate a joint schedule with minimal length.

5. *Chapter 6.* This chapter presents the conclusions, and provides a summary of research outcomes and future research directions.

Chapter 2

Literature Review

This chapter reviews link schedulers for TDMA-based WMNs, solutions to the personalized broadcast or data collection problem, load balance routing methods, and link scheduling approaches for full-duplex networks.

2.1 TDMA WMNs with Directional Antennas

To date, directional antennas have been widely used in WMNs, see for example [33] [6] and [34], because they provide longer transmission and reception range, higher gain, better spatial reuse and minimize interference [35]. To this end, a number of link scheduling works seek to exploit these characteristics to maximize spatial reuse or network capacity. Note, for solutions that assume CSMA, the reader is referred to [36]. In the following subsections, link scheduling works are categorized into three different groups according to antenna type; namely, single-radio, multi-radio and smart antennas.

2.1.1 Single-Radio

Past works have considered equipping nodes with an electronically steerable antenna that allows them to transmit/receive from a specific direction. In this setting, the

main aims or problems include maximizing network capacity, satisfying the transmission requirement of each link, minimizing end-to-end delays and guaranteeing end-to-end throughput.

Liu et al. [37] propose a heuristic algorithm to maximize network capacity. The algorithm is executed at the beginning of each time slot by a central controller. At the start of time slot t, the controller orders unscheduled transmissions according to their arrival time; i.e., in a first-in-first-out (FIFO) manner. The controller then greedily assigns unscheduled, non-interfering transmissions into slot t.

In [38], the authors propose a TDMA based MAC protocol to maximize network capacity. Each node is equipped with one switch beam antenna that works either in omni or directional mode. A node transmits control messages omni-directionally and uses directional mode for data transmissions. Each node has a unique ID ranging from 1 to N, where N is the number of nodes. Each node maintains an information table containing the ID and coordinates/location of all nodes. Time is divided into slots and each superframe contains N slots. A node with ID n is called the *Main* node of the n-th slot. Other nodes are called *Normal*. Each time slot is further divided into a control and data part. The control part consists of one broadcast period followed by two contention periods. During the broadcast period, the *Main* node sends out a notify message omni-directionally to inform its neighbors that it has a pending data packet. The notify message contains the ID and location of the source and destination node. Each *Normal* node that has a data packet to send first checks whether it will interfere with existing packet transmissions. Non-interfering nodes contend for the channel using the binary countdown protocol of [39]. Specifically, each node creates a unique binary sequence from its priority and ID. A node's priority is determined by the urgency and waiting time of its buffered packets. Each node contending for the channel sends its binary sequence bit by bit. The binary bits sent by each node are OR-ed together by the channel. Once a contending node notices that its zero bit becomes a one, it quits channel contention. The winning node sends out a notify message omni-directionally. This notify message contains

the ID and location of the source and destination node of a packet transmission. Packets are transmitted in the data part of each time slot directionally.

The authors in [40] propose a slot borrowing scheme. Nodes send control messages omni-directionally and data packets are sent directionally. Their scheme assumes a TDMA link schedule and aims to utilize unused time slots to increase transmission opportunities. Each node knows the number of packets buffered at its neighbors. At the beginning of each time slot, each node first checks for an on-going transmission. If there is no packet transmission, a node initiates a slot borrowing scheme. Specifically, the node with the highest queue length notifies its pending packet transmission to its neighbors by sending a notification message omni-directionally. The notification also contains the location of the target neighbor of its head of line packet. Then the node with the second highest queue length checks whether it is able to transmit one of its packets without interfering with the node that has the highest queue length. If it is able do so, it then broadcasts a notification message to its neighbors. After that, the node with the third highest queue length checks whether it is able to transmit, and so forth.

Sanchez et al. [41] propose a centralized greedy algorithm called GreedyRS that aims to maximize the number of simultaneous transmissions in each slot k. In addition, the number of slots assigned to each link is proportional to its traffic load. Each link l has priority M_l, which is initially set to the number of slots required by the link. GreedyRS first sorts all links in decreasing priority order. Then GreedyRS checks each link, starting with the one that has the highest priority, and schedules a link into slot k if it does not interfere with links that are already scheduled in slot k. After scheduling all possible links in slot k, GreedyRS reduces the priority of all links scheduled in slot k by one. GreedyRS stops when M_l for each link l is zero.

The authors of [42] extended their previous work in [41] and propose a max-min fair link scheduler for WMNs with directional antennas. Specifically, they aim to maximize the minimum source rate of all traffic flows. To do this, they formulate the link scheduling problem as an integer linear program (ILP). They then apply

the column generation method to find the optimal link schedule. The authors also proposed a heuristic approach because the number of decision variables grows exponentially with the number of links. The key idea is to assign time slots to each link based on traffic load. In addition, the heuristic aims to separate assigned slots uniformly along the frame to reduce queueing time. Their proposed algorithm maintains two link lists: L_A and L_B. Each link has priority p. Initially, all links are in L_A and have priority $p = 0$. A link will be moved to L_B if it has sufficient slots assigned to it. To schedule links activated in slot k, the algorithm first sorts all links in L_A and L_B in decreasing priority order. Then the algorithm checks all links in L_A starting with the link that has the highest priority and schedules a link into slot k if it does not cause interference. After scheduling all possible links in L_A, the algorithm checks whether any links in L_B can be activated in slot k to further increase capacity. To do this, the proposed algorithm checks all links in L_B in decreasing priority order, and activates a link in slot k if it does not cause interference. After scheduling all possible links in slot k, the priority of links in slot k is set to zero while the priority of all other links is increased by a factor that corresponds to their traffic load. Their algorithm terminates when L_A is empty.

Panigrahi et al. [43] aim to minimize end-to-end delays. They consider the network model proposed in [44]; each node is equipped with a high-gain directional antenna to provide long distance transmission. Their network topology is a tree with two levels and rooted at a central node with wired connectivity to the Internet. In addition, the authors consider only packets flow from a root node to other nodes. As the authors consider a tree with two levels, each flow has only two hops. To minimize delay, their algorithm aims to compute the shortest superframe. In addition, for each flow, the link between the root node and level-1 nodes activates before the links connecting level 1 and 2 nodes. Their proposed algorithm is based on graph coloring, where interfering links are assigned different colors and are activated in different time slots. Specifically, the authors use four simple *reference* topologies. Each reference topology contains one or more pairs of non-interfering flows. They

then color each one by greedily assigning one color to each link. Then their proposed algorithm divides the network into small pieces where each piece matches one of the four reference topologies. To do this, their algorithm repeatedly picks non-interfering flow pairs according to these reference topologies. Once one or more pairs of flows are picked, their links are assigned a color according to the coloring result of the matched reference topology. Colored flows are then removed from the network, and the algorithm stops when each flow has a color.

Cain et al. [45] propose a distributed link scheduling algorithm to meet different transmission demands of each flow by assigning time slots dynamically. All data transmissions are directional whilst control information is sent omni-directionally using a separate channel. In their approach, each superframe is divided into two parts: semi-permanent (SP) and demand-assigned (DA). When a new node joins, it establishes a link with its neighbors by exchanging control messages over its omni-directional antenna. Then for each neighbor, the new node assigns one slot in the SP part so that a node has guaranteed capacity to each neighbor. If a single slot in the SP part cannot satisfy the traffic load of a link, the sender node sends a request to the intended receiver requesting for more transmission slots. The request message contains the sender's free slot in the DA part. If the receiver has sufficient free DA slots to receive from the sender, the receiver sends an accept message. The sender then returns a confirm message.

Zhang et al. in [13] and [46] propose a distributed TDMA-based algorithm that uses directional transmission and reception to meet link demands. Time is divided into frames and each frame is divided into three sub-frames. The slots in the first and second sub-frame are further divided into mini-slots. Neighbor discovery is performed in the first sub-frame. Each node discovers its neighbors by steering its directional antenna to scan and establish a connection with its neighbors using a three-way handshake. Each pair of nodes exchange their available slots in the second sub-frame and reserve the first common free slot for future message exchanges. For each pair of nodes, the node that initializes the three-way handshake in the first

sub-frame, say node A, will transmit first in the reserved time slot in the second sub-frame. During the second sub-frame, node A first sends a message M_1 containing the number of slots it requires for data transmissions and its available time slots in the third sub-frame. A node B, which receives the M_1 message, compares its free slots with node A's available slots. If node B has sufficient slots that can be used to receive, it replies with message M_2 to node A by reserving the first k common free slots for data transmissions. Here k is the number of slots required. Otherwise, if node B does not have enough free slots, node B will send back a reject message. Node A sends an acknowledgment M_3 to node B after it receives M_2 successfully. The third sub-frame is used for data transmissions.

The authors in [47] propose an integrated neighbor discovery and MAC protocol named PMAC. Each node is equipped with a single steerable directional antenna. PMAC relies on a superframe that is divided into three phases: search, poll and data transfer. Each phase contains several time slots. The search phase is used by nodes to find new neighbors whereby a node steers its antenna in a randomly chosen direction. Each slot in the search phase can be further divided into four sub-slots. The first and second sub-slots are used for nodes to search and establish a connection with their neighbors. If a node successfully establishes a new connection, it and its neighbor use the third and the fourth mini-slot in the search phase to reserve one slot in the poll phase. Specifically, each pair of nodes exchange their free slots in the poll phase and reserve the first common slot in the poll phase. In the poll phase, each slot is also further divided into four mini-slots. The first two mini-slots are used for each pair of nodes to re-establish a connection. The third and fourth mini-slots are used for each pair of nodes to exchange their transmission requirement and reserves slots for packet transmission in the data transfer phase. To do this, each pair of nodes inform each other the next packet they need to send and the number of required time slots. Each node also indicates its available time slots that it is able to transmit to or receive from the corresponding neighbor. Packets are transmitted in the data transfer phase.

To date, there are several works that consider scheduling 60 GHz or mmWave links; e.g., [48] [49] [50] and [51]. In particular, they seek centralized algorithms that take advantage of the following key characteristics of the 60 GHz band: high propagation loss and low multi-user interference (MUI). These characteristics enable concurrent transmissions and thus increase network capacity.

The authors in [48] propose a MAC protocol to maximize network capacity. The network consists of several wireless nodes and a single piconet controller (PNC) where each node is equipped with a directional antenna. Each superframe is divided into three phases: beacon period (BP), contention access period (CAP) and channel time allocation period (CTAP). Specifically, BP is used for network synchronization and control messages. CAP is used by nodes to send transmission requests to the PNC. Data is transmitted during the CTAP. The PNC collects the number of slots required by each node during the CAP. The PNC goes through each link and greedily assigns a link to the earliest available time slot in the CTAP if it does not interfere with links already assigned in that slot. After generating the schedule, the PNC informs each node its assigned slots in the CTAP.

Son et al. [49] propose a frame-based scheduling protocol to maximize network capacity. Each superframe is divided into four phases: poll (T_{poll}), scheduling (T_{sch}), push (T_{push}) and transmission (T_{tr}). Specifically, during T_{poll}, the PNC polls each node to collect transmission requests. The T_{sch} phase is used by the PNC to compute a schedule. The PNC then sends the generated schedule to nodes during T_{push}, and T_{tr} is used for data transmissions. To generate a schedule, the protocol generates a directed graph with weighted links that correspond to a link's transmission demand. They then use a greedy edge coloring algorithm, whereby all links are traversed in decreasing weight order, and adjacent links are assigned a different color. Links with the same color are scheduled in the same time slot.

Qiao et al. [50] aim to increase the network capacity of 60 GHz wireless networks. In their proposed approach, the superframe is divided into three parts: beacon period (BP), contention access period (CAP) and channel time allocation

period (CTAP). Briefly, BP is used for network synchronization and control messages, CAP is used by nodes to send transmission requests to the PNC, and CTAP is used for data transmissions. The scheduling problem is to determine the maximum number of flows that can be scheduled within the CTAP while satisfying the minimum throughput requirement of each flow. To maximize the number of flows, the PNC repeatedly generates a link activation set that contains links that can transmit simultaneously. Links contained in the link activation set U_i will be activated in the i-th slot in the CTAP. To generate U_i, the PNC checks each link, and adds a link to U_i if it does not interfere with other links that are already contained in U_i. In addition, once a link is added into the activation set, it will remain in the set until its transmission demand is satisfied. Their algorithm stops when each link has minimum throughput.

In [51], the authors present a multi-hop concurrent transmission scheme to maximize network capacity. The proposed scheme consists of a hop selection metric to select relays for each session/flow and a link scheduling approach to allocate active time for the links of each flow. To select relay nodes, the PNC generates a weighted graph. Each link has an associated weight that corresponds to the link length and traffic load. Specifically, the link weight between node A and B is given by $w_{A,B} = (\frac{d_{A,B}}{\overline{d}})^n + \frac{F_B}{\overline{F}}$. Here $d_{A,B}$ is the Euclidean length of the link between A and B, $\overline{d}$ is the average Euclidean link length, F_B is the traffic load of node B and $\overline{F}$ is the average node traffic load. For each flow, their hop selection method aims to minimize the total aggregated link weight using the Dijkstra algorithm. To schedule the links of each flow, their proposed link scheduling approach first sorts all links in decreasing order according to link weight. They then adopt the approach in [50] to repeatedly generate link activation sets. This approach checks each link starting with the highest weighted link. A link is added into a link activation set if it does not cause interference with links that have already been added. Links in link activation set U_i will be activated in the i-th slot.

Recently, there are several cross layer approaches that jointly consider TDMA

link scheduling together with routing, e.g., [52] and [53], or power control [54] [55]. The authors of [52] jointly consider routing and link scheduling to minimize the superframe length. They formulate the routing and link scheduling problem as a mixed integer linear program (MILP). The decision variables are the transmission power of each link and the traffic routed over each link. The MILP is solved using column generation. In [53], the authors aim to provide guaranteed end-to-end bandwidth by jointly considering routing and link scheduling. Specifically, each link is associated with a weight determined by the number of interfering links and traffic load. The key idea is to run the Bellman-Ford algorithm to find the shortest path. Links along the shortest path will experience minimal interference. In addition, each link is assigned slots to meet its traffic demand. The authors of [54] jointly consider link scheduling and transmission power control to maximize network throughput. They formulate the problem into an MILP where they use the transmission power of each link and whether a link is scheduled as decision variables. The number of decision variables grows exponentially with the number of links and thus the authors propose a heuristic algorithm to solve the link scheduling and power control problem. The algorithm first constructs a conflict graph based on link interference. It then greedily schedules as many links as possible into each time slot to maximize network capacity. In [55], the authors aim to minimize the total energy consumption for all traffic flows by jointly considering routing and scheduling. They use Dijkstra's algorithm to find the minimum cost path for each packet. The link cost is determined by the energy consumed in transmitting and receiving one packet over the link. To schedule links, they first generate a weighted graph, where the weight of each link is associated with the number of packets flowing through it. They then repeatedly find the maximum weighted matching of a graph and assign time slots to each link accordingly.

Table 2.1 summarizes the TDMA link scheduling algorithms reviewed thus far. References [37] [38] [40] [41] [48] and [49] aim to maximize capacity. Specifically, they increase network capacity by maximizing concurrent transmission pairs. Among

works that maximize capacity, references [38] and [40] are distributed solutions. Distributed approaches only require local information and their computation time does not increase with network size. However, in centralized approaches, the computation time increases exponentially with the network size. Moreover, only [41] [42] [43] [51] [52] [53] [54] and [55] consider multi-hop networks while other works consider single hop packet transmissions. In addition, all reviewed works have only considered nodes with a single directional antenna.

Work	**Centralized or Distributed**	**Transmission Mode**	**Channel**	**Solution**
Liu et al. [37]	Centralized	Single-transmit-receive	Single	Schedule transmissions in a FIFO manner starting from the earliest unscheduled one
Hao et al.[38]	Distributed	Single-transmit-receive	Single	Node with ID n transmits in the n-th slot
Paso et al. [40]	Distributed	Single-transmit-receive	Single	Nodes with the highest queue length coordinate with their neighbors to utilize unused time slots
Sanchez et al.[41]	Centralized	Single-transmit-receive	Single	Always schedule the link requiring the most number of slots
Garache et al.[42]	Centralized	Single-transmit-receive	Single	Use ILP to maximize the minimum flow rate, and employ a heuristic to schedule links according to link load
Panigrahi et al.[43]	Centralized	Single-transmit-receive	Single	Use edge coloring to schedule four *reference* topologies. Divide the network into small pieces to match a *reference* topology
Cain et al. [45]	Distributed	Single-transmit-receive	Two	Assign links with one slot to guarantee capacity. Then assign extra slots based on link load
Zhang et al. [13] [46]	Distributed	Single-transmit-receive	Single	Each node steers its antenna to discover neighbors. Assigns time slots according to link load
Jakllari et al. [47]	Distributed	Single-transmit-receive	Single	Each pair of nodes exchange their transmission requirement and reserve the first available time slot to transmit/receive
An et al.[48]	Centralized	Single-transmit-receive	Single	Assigns the earliest available time slot to a link

Son et al.[49]	Centralized	Single-transmit-receive	Single	Generate a directed weighted graph based on transmission demand. Color the graph in decreasing weight order
Qiao et al. [50]	Centralized	Single-transmit-receive	Single	Repeatedly adds a link into transmission set. A node will remain in the generated set until its transmission requirement is met
Qiao et al. [51]	Centralized	Single-transmit-receive	Single	Select a relay node according to link length and load. Then use the method in [50] to add links into concurrent transmission set according to link weight
Capone et al. [52]	Centralized	Single-transmit-receive	Single	Formulate the joint routing and scheduling problem as an MILP and solve using column generation
Lu et al. [53]	Centralized	Single-transmit-receive	Single	Generate a weighted graph using link load. Use Bellman-Ford algorithm for routing and schedule links in decreasing weight order
Ramamurthi et al. [54]	Centralized	Single-transmit-receive	Single	Use MILP to solve the joint problem. Also use a heuristic to schedule links according to transmission requirement
Spyropoulos et al. [55]	Centralized	Single-transmit-receive	Single	Generate a weighted graph according to link load. Then repeatedly find the maximum matching and assign time slots according to link load

Table 2.1: Comparison of works where nodes have a single directional antenna

2.1.2 Multiple Radios

Nodes in WMNs can also be equipped with multiple directional antennas. They thus have the potential of transmit to or receive from multiple neighbors simultaneously, aka MTR capability. This further increases network capacity. For example, in [6], the authors expound a real world implementation of a low-cost WMN that is used to provide connectivity to rural villages. In particular, each node uses off-the-shelf IEEE802.11 radios and is equipped with multiple high-gain directional antennas to achieve MTR. To take advantage of the MTR capability nodes, the authors introduce 2P, a centralized MAC protocol to maximize network capacity. The 2P MAC operates by switching each node between two phases: SynRx and SynTx. If a node is transmitting on all links (SynTx), all of its neighbors must be in reception mode (SynRx), meaning the topology must be bipartite.

Chin et al. [30] extended Raman's work [6] to address a key limitation of 2P; that is, the network topology must be bipartite. The authors propose a heuristic that creates a MAX-CUT [29] in each time slot. Their proposed solution, named Algo-1, does not require a bipartite topology. Instead, in each time slot, the topology is divided into two maximally connected sub-graphs. To create sub graphs, Algo-1 recursively divides the topology into two disjoint, maximally connected sets. Nodes transmit in a time slot and become receivers in the following time slot. To construct these sets, Algo-1 places all nodes in one set. Algo-1 then moves a node to the other set if the total connections to nodes in the same set is higher than to those in the other set. In addition, Algo-1 adds opportunistic links to a generated schedule. Specifically, opportunistic links are those links that have been activated in prior slots. However, in [30] the authors do not consider different link loads. This limitation is addressed in their subsequent work [56], where they modify Algo-1 to consider weighted links.

In [31], the authors extended the approach in [30] and [56] by proposing a link scheduling algorithm called Algo-2 that computes a new MAX-CUT in each slot,

as opposed to every other slot. To generate a new MAX-CUT, Algo-2 creates two node sets and places all nodes in one set. Similar to Algo-1, Algo-2 moves a node to the other set if it has more connections with nodes in the same set than those in the other set. In addition, the authors in [31] consider delays. Specifically, they propose a heuristic called Bucket Draining Algorithm (BDA) to further reduce transmission delays. Specifically, BDA reorders some slots in superframe S to achieve better delay. BDA first sorts links in decreasing link weight order and creates a new superframe R. The weight of a link is the number of required slots. BDA then repeatedly picks a slot in S that contains the highest weighted link in S, moves it to R and removes it from S. BDA stops when S is empty.

Nedevschi et al. [25] present a distributed version of the 2P MAC, named JazzyMAC. They address the following limitations of 2P: (i) the network topology must be bipartite, (ii) fixed length transmission slots, meaning 2P cannot adapt to dynamic traffic loads. Specifically, JazzyMAC works as follows. Each link is associated with a token, where only the node holding a token can transmit on the associated link. The basic operation of JazzyMAC is governed by the following rules: after a node has finished transmission on a link, it computes a time value in which it is ready for reception. The node then sends the token together with the computed time value to the other end of the link. A node becomes a transmitter if it holds the token for all its links. In other words, a node in transmit mode changes to the receive mode when it releases all tokens. A node can transmit on a link only when it is in transmission mode and the token for a given link is valid; i.e., the corresponding end node is ready for transmission. A key consideration is the initial token assignment. If tokens are assigned improperly to nodes, a WMN will experience deadlock or poor link utilization. Henceforth, the authors first use graph coloring to ensure no two adjacent nodes have the same color. Then for each link, the token is assigned to the end node that has the lowest color.

In [34], the authors propose a centralized link scheduling algorithm to maximize capacity. Their proposed algorithm greedily maximizes the number of links

scheduled in each time slot. To do this, their algorithm first creates two link sets: *assigned* and *unassigned*. Initially, all links are in the *unassigned* set. Once a link is assigned a time slot, it is moved to the *assigned* set. To schedule links in slot k, their algorithm goes through each link in the *unassigned* set, and schedules as many links in slot k as possible if the scheduled links do not interfere with each other. Their algorithm stops when each link is assigned one time slot.

In [57], the authors propose a centralized scheduling algorithm to meet traffic demands with the minimum transmission time for MTR WMNs. They first formulate the multi-transmit-receive problem as maximizing the number of simultaneous transmission pairs; i.e., maximum matching. They then propose two heuristic algorithms called Heavy-Weighted-First (HWF) and Max-Degree-First (MDF). HWF repeatedly constructs a maximum matching set according to the traffic demand of each link. This is performed by a central controller whereby it traverses each link starting with the highest demand. The controller then adds a link into a matching set if the link does not interfere with those already in the set. Upon completion of each round, the central controller assigns time slots to the constructed matching set such that at least one link's demand is met. MDF works essentially the same as HWF. Specifically, MDF first constructs a conflict graph. It then goes through each vertex in the conflict graph, i.e., a link in the network, in decreasing node degree order, and adds it into a matching set if it does not interfere with links that already in the set.

Rhee et al. [58] present a distributed randomized time slot scheduling algorithm, called DRAND, to maximize capacity. In DRAND, each node sets itself a probability to broadcast a transmission request to its neighbors at the beginning of each superframe. For a node n, its probability is set to $\frac{1}{k}$, where k is equal to the number of one-hop and two-hop neighbors of node n that have not been assigned a slot. A node v that receives a transmission request from a neighbor u will reply with a grant message containing node v's free slots. After receiving a grant message, node u compares its own free slots with those specified in the grant message. If node u

has free slots, it schedules a transmission from v to u in its free slots. Node u then broadcasts a release message containing its busy slots and its intended receivers to its neighbors.

Das et al. [59] propose a distributed link scheduling algorithm, called RTDMA-DA. The authors assume a rectangular grid topology, where a node is placed at each intersection. Each node is equipped with four directional antennas, each facing a neighbor and is associated with integer coordinates. There are two frame types: reservation frame (RF) and information frame (IF). Both type has equal length. Each RF is followed by an IF, where RF is used to generate the schedule while IF is used for data transmission. Each RF consists of several reservation slots. Each of these slots has two reservation cycles. Each node holds a binary variable called *canRes*. A node is able to send a Reservation Packet (RP) to its neighbors during a reservation cycle if *canRes* has a value of one. Otherwise, it listens for RPs from its neighbors. The *canRes* value of each node is initialized at the start of each reservation slot as follows. If the x and y coordinate of a node are both even or odd, its initial *canRes* value is one. Otherwise, its *canRes* value is set to zero. This ensures the *canRes* value of a node is different to that of its neighbors. In addition, a node that has sent reservation packets in the first reservation cycle listens to its neighbors in the second reservation cycle, and vice-versa. RTDMA-DA generates a link schedule as follows. In the first reservation cycle, each node first checks whether it has data packets to be transmitted and whether its *canRes* value is one. If both have a value of one, a node sends out a reservation message to neighbors with pending packets. In the second reservation cycle, each node that has not received a reservation message in the first reservation cycle checks whether it has any buffered packets. If yes, it sends out a reservation message to the corresponding neighbors that have not previously sent it a reservation message. Packets scheduled in the i-th reservation slot of the current reservation frame will be transmitted in the i-th slot in the following information frame.

The authors in [60], [61] and [62] jointly consider routing and link scheduling

in TDMA WMNs with directional antennas. In [60], the authors aim to maximize network capacity. To do this, they formulate the joint routing and link scheduling problem as an ILP to assign slots to meet link demands. In [61], the authors aim to maximize the number of concurrent flows using a joint routing and scheduling scheme. They first formulate the routing problem as an LP to maximize the number of concurrent flows. They then propose a novel link scheduling algorithm, called Multi-DEC, to generate the link schedule. Specifically, Multi-DEC first generates a weighted graph where the weight of each link corresponds to the number of flows that passes through it. Next, the algorithm constructs a multi-graph based on the weighted graph where each link e with a weight of w_e is replaced with w_e parallel, directed links; each with a weight of one. The algorithm then splits the multi-graph into several simple sub-graphs where there is only one direct link between each pair of nodes. Each sub-graph is then colored using DEC [63]. DEC works according to the following rules. It first determines the chromatic number ω of a given graph G. Here, the chromatic number of a graph G is the smallest number of colors needed to color all the vertices of G so that no two neighboring vertices share the same color. Then, DEC finds the minimum ϖ that satisfies $\binom{\varpi}{\lfloor \varpi/2 \rfloor} \geq \omega$. After that, for each vertex, DEC assigns them the same set of $\lfloor \frac{\varpi}{2} \rfloor$ *edge* colors. By definition, the condition $\binom{\varpi}{\lfloor \frac{\varpi}{2} \rfloor} \geq \omega$ ensures that there are sufficient *edge* color sets that can be assigned to each vertex. Finally, for each *directed* edge, DEC assigns one color that is in the *edge* color set of the transmit end but not in the *edge* color set of the receive end. In a different work, Wang et al. [62] also consider end-to-end delays when generating a schedule. They propose two joint routing and link scheduling solutions: JRS-BIP and JRS-Multi-DEC respectively. Specifically, both solutions use the algorithm proposed in [31] to generate a link schedule. The main difference is how the routing path is chosen. In JRS-BIP, they formulate the routing problem into a binary linear program (BIP). Specifically, for each source-destination flow, JRS-BIP chooses k shortest paths from the source to destination. The path to be chosen is set as a decision variable. In JRS-Multi-DEC, the authors first use the

link coloring approach proposed in [61] to color each link. Then for each source-destination flow, JRS-Multi-DEC goes through its k shortest paths and chooses the path with the lowest weight. Here the weight of a path is calculated by number of hops times the number of colors used along the path.

As mentioned in Chapter 1, MTR can also be realized by equipping nodes with multiple 60 GHz directional antennas. In [64], the authors aim to maximize network throughput by jointly considering routing and link scheduling in 60 GHz WMNs. Specifically, the authors formulate the routing problem into an ILP, where the number of flows across each link is used as decision variables. They then propose a heuristic algorithm to generate the link schedule. To do this, they first sort all flows in decreasing order according to the load of each flow. Then their algorithm picks the flow with the highest load and assigns increasing time slots to links along the flow from the source to destination node.

The multiple radios works reviewed thus far assume all data packets are transmitted over the same channel. Hence, interfering links must be assigned to different time slots to avoid collision. However, researchers have also considered assigning different radios to work on different channels to increase spatial reuse. In this case, interfering packets are transmitted in different orthogonal channels to eliminate interference [63]. Readers are referred to [65] and [66] for more multi-channel multi-radio works. It is worth noting that these works are outside the scope of this thesis. In particular, this thesis only considers MTR over a single channel.

Table 2.2 summarizes link scheduling approaches that consider multiple directional antennas. The approaches proposed in [6] [30] [56] [31] [34] [58] [59] and [64] aim to maximize capacity. References [6] and [58] require bipartite and grid topologies respectively, while other approaches do not have this limitation. In addition, only [25], [58] and [59] are distributed solutions. Moreover, all works reviewed in this section increases spatial use by pointing a node's transmission beam towards intended receivers and no work has considered using antenna elements to suppress interference. References [57], [58] and [61] consider single hop networks while other

works consider multi-hop packet transmissions.

2.1.3 Smart Antennas

Smart antenna systems have similar properties to directional antennas. Each smart antenna system consists of multiple antenna elements at the transmitting and/or receiving side of the communication link. These antenna elements thus allow a transmitter or/and receiver to exploit the spatial dimension of the wireless channel. Moreover, these antenna elements enable (i) spatial diversity, which helps increase signal quality, and (ii) spatial multiplexing, which allow the transmission/reception of multiple independent data streams [7]. In addition, nodes are able to suppress interference or null their transmissions [67].

Bao et al. [24] propose a distributed link scheduling protocol called Receiver-Oriented Multiple Access (ROMA) for multi-hop wireless networks that employ smart antennas. Each node is equipped with a multi-beam adaptive array (MBAA) antenna with K beams so that each node is able to transmit to or receive from up to K neighbors simultaneously. ROMA schedules links in each time slot using a hash function to determine the set of transmitting nodes. Here, the hash function is a fast pseudo-random number generator that produces an unsigned integer message digest of the input bit stream. In each time slot, a node determines its priority using a hash function, which takes as inputs the node ID and current time slot number. In a given slot, a node becomes a transmitter if its priority value is odd, else it is a receiver. If a node and its entire one-hop neighbors are all transmitters or receivers, the node enters the opposite mode. A node also uses the hash function to determine the priority of all its links. Here the priority of a link is determined by hashing the ID of end nodes and link weight. In each time slot, each receiving node activates its highest K links.

In [68], the authors developed a distributed algorithm to maximize network capacity. They assume a node can rapidly switch between transmit and receive mode

Work	Centralized or Distributed	Transmission Mode	Objective	Solution
Raman et al. [6]	Centralized	Multi-Transmit-Receive	Provide rural connectivity	Each node switches between SynTx and SynRx mode
Chin et al. [30]	Centralized	Multi-Transmit-Receive	Maximize capacity	Generate a MAX-CUT in every other slot
Chin et al. [56]	Centralized	Multi-Transmit-Receive	Maximize capacity	Generate a MAX-CUT in every other slot
Loo et al. [31]	Centralized	Multi-Transmit-Receive	Maximize capacity	Generate a new MAX-CUT in each time slot
Nedevschi et al. [25]	Distributed	Multi-Transmit-Receive	Meet dynamic traffic load requirement	A node holding the token of all its links transmit
Hazra et al. [34]	Centralized	Multi-Transmit-Receive	Maximize capacity	Use a greedy method that goes through all unassigned links and schedule a link intp a slot if it does not interfere with other links in the slot
Dai et al. [57]	Centralized	Multi-Transmit-Receive	Meet traffic demand using the shortest superframe	Schedule link with the highest weight first and maximum degree first
Rhee et al. [58]	Distributed	Multi-Transmit-Receive	Maximize capacity	Each node has a probability to request slots for transmission
Das et al. [59]	Distributed	Multi-Transmit-Receive	Maximize capacity	Nodes send out reservation messages according to their coordinates
Crichigno et al. [60]	Centralized	Multi-Transmit-Receive	Maximize capacity	Use ILP to solve a joint routing and scheduling problem
Dutta et al. [61]	Centralized	Multi-Transmit-Receive	Maximize number of concurrent flows	Use LP to generate routing paths and edge coloring to schedule links
Luyao et al. [62]	Centralized	Multi-Transmit-Receive	Minimize end to end delay	Use BIP and Multi-DEC [61] to generate routing paths. Use [31] for link scheduling
Su et al. [64]	Centralized	Multi-Transmit-Receive	Maximize capacity	Use ILP to generate routing paths and schedule flows in decreasing load order

Table 2.2: Comparison of works using multi-radio directional antennas

so that it can respond to requests and receive responses quickly. Moreover, a node listens omni-directionally for transmissions when it is idle. Each frame is divided into a reservation and data transmission part. At the beginning of each frame, a node that has data to send randomly selects a slot in the reservation part and sends one or more request messages to its intended receiver(s). The request message contains the number of slots needed to transmit data packets and its available slots in the data transmission part. After sending out request messages, the sender quickly switches to receive mode to await respond messages. A node that successfully receives a request message compares its own free slots with the free slots of the sender. The receiver then selects the required number of time slots starting from the first overlapping time slot and informs the sender of the selected slots.

The authors in [69] aim to maximize the number of concurrent links in each slot. Each node is equipped with digital adaptive arrays. Nodes are able to use their antenna elements to transmit/receive or interference cancellation. Their proposed link scheduling approach is based on link coloring. To assign a color to each link, their algorithm first sorts all nodes in decreasing order according to the number of neighbors. Then their algorithm repeatedly picks the node with the highest number of neighbors and assigns a different color to its links. Links assigned with the same color activate in the same slot. Their algorithm stops when all links are assigned a color.

The authors in [70] aim to increase capacity by maximizing concurrent streams in each time slot. Each stream is associated with a priority, which is determined by data type and delay. To schedule streams in slot t, their proposed algorithm first sorts all streams in decreasing priority order. Then their algorithm assigns as many concurrent streams to slot t as possible if the interference caused by these streams can be suppressed successfully.

In [71], the authors propose a distributed link scheduling algorithm to maximize network capacity. Their algorithm assumes nodes are organized into clusters, where each cluster consists of a head node and several member nodes. They assume each

node is equipped with a k beams smart antenna system and is able to transmit to or receive from multiple neighbors simultaneously. They also assume each node knows the location of its neighbors. Their proposed approach can be divided into three phases. In the first phase, the head node collects the transmission requirement of each member node by sending out a polling message on all its k beams. During the second phase, their algorithm constructs parallel transmission sets with the maximum number of links, where links in the same set do not interfere with each other and are able to activate simultaneously. The third phase is used for data transmission. Specifically, there are two types of parallel transmission sets: Host-centric Parallel Sets (HPSs) and Cluster-centric Parallel Sets (CPSs). During the second phase, the cluster head first constructs several HPSs for each node. It then constructs several CPSs for the cluster, where each CPS is a collection of HPSs. To construct HPSs for a member node, the cluster head first constructs several empty HPSs. Then the cluster head greedily adds one link of the member node to an HPS if the link does not interfere with the links that exist in HPS. To construct CPSs, the cluster head first constructs several empty CPSs. Then the cluster head greedily adds one HPS into a CPS while ensuring all links in a CPS do not interfere with each other. Finally, the cluster head sorts all CPSs in decreasing order according to the number of links in CPS and assigns a time slot to each CPS starting from the CPS with the biggest size.

Hung et al.'s work [72] extends that of [71] to consider packet delays. They adopt the algorithm proposed in [71]. The difference between their approach and [71] is as follows. When the cluster head assigns slots to CPSs, the cluster head sorts CPSs in decreasing order according to the delay of links in each CPS. However, in [71], the cluster head sorts CPSs according to the number of links in each CPS. The cluster then repeatedly picks the CPS that contains the link with the smallest delay and assigns one slot to that CPS.

Jawhar et al. [73] develop a distributed protocol to provide guaranteed bandwidth between any source and destination pairs. Each node is equipped with a beam-

forming antenna system that is able to work in both directional and omni-directional mode. Control information is exchanged omni-directionally using a separate channel. The key idea is that each source node broadcasts a reservation packet to find the shortest path to the destination node. The reservation packet flows through the network and records all nodes along its path. Nodes along the path reserve sufficient time slots to relay the packet. Specifically, when a node receives a reservation packet, it first checks whether it is the destination node of the reservation packet. If not, it reserves the required slots to transmit the packet indicated by the reservation packet. Then the node records itself as an intermediate node of the path in the reservation packet together with its reserved slots. The intermediate node then sends the reservation packet to its neighbors. Note, a node will drop a reservation packet if it is already an intermediate node contained in the packet. Lastly, a destination node sends a message along the path recorded in the reservation packet to confirm the reservation.

In [74], the authors aim to minimize superframe length. Their proposed algorithm first generates a conflict graph, where each vertex is a link in the network graph. It first sorts all vertices in the conflict graph in decreasing vertex degree order. Then the algorithm repeatedly picks the vertex that has the maximum node degree and assigns it a color while ensuring each vertex does not have the same color with its neighbors. The authors aim to use the minimum number of color to get a shorter superframe. Specifically, their algorithm first checks whether a color assigned to a different vertex can be used before picking a new color.

In [75], the authors propose a link scheduling algorithm to meet the transmission demand of each link. Their network consists of a central controller and several static nodes. They assume nodes in the network are synchronized and are able to receive scheduling information from the controller. Each superframe is divided into link allocation and data transmission period. During the link allocation period, their proposed algorithm first sorts all links in decreasing order according to their demand or the number of required slots. To generate the link schedule for slot

k, their algorithm goes through each link starting with the link with the highest demand. A link is scheduled in slot k if it does not cause interference to links that are already scheduled in slot k. After generating the schedule for slot k, the demand of scheduled links is reduced by one. Their algorithm stops when each link is assigned sufficient number of slots.

To date, there are several cross layer approaches that jointly consider TDMA link scheduling and routing in WMNs with smart antennas; e.g., [76] and [77]. Specifically, the authors in [76] assume each node is only able to transmit one packet at a time while its other antenna elements are used to suppress interference. They formulate the joint routing and link scheduling problem as an LP, which determines the number of flows crossing a link and whether a link is active. One limitation is that the LP solution may not be integers; i.e., the LP yields an infeasible link schedule. To address the said limitation, the authors propose a heuristic algorithm. Specifically, the authors first round up the number of flows crossing each link into an integer. The authors then use this rounded value as the link weight to create a multi-graph. The authors then greedily assign each link into the first available and non-interfering slot. In [77], the authors aim to maximize network throughput. They assume there are multiple routing paths between the source and destination node of each flow. In their proposed approach, they model the multi-path routing problem as an MILP to determine the active time of each link. They then propose a heuristic link scheduling algorithm to generate the link schedule. They first divide links into N different link groups, where N is the number of nodes. To do this, their algorithm goes through each node, and groups all its incoming links into one link group. This means each link group contains all incoming links of a node. Then they generate a conflict graph, where each vertex is one link group. Lastly, their algorithm traverses each vertex of the conflict graph, and assigns a different color to each vertex.

Table 2.3 summarizes the aforementioned link scheduling approaches for WMNs with smart antennas. The approaches outlined in [76] and [74] assume nodes use

their antenna elements to suppress interference in order to increase spatial reuse. However, other approaches increase spatial reuse by forming transmission beam towards receivers and reduce the impact of side lobes. The authors in [73], [76] and [77] consider multi-hop packet transmissions while other works only consider one hop packet flow.

2.2 Personalized Broadcast and Data Collection Scheduling

This section focuses on recent approaches that address the personalized broadcast or the equivalent data collection problem. Note that there are many existing approaches, e.g., [78], that focus on energy consumption, reliability and quality of services (QoS). This section, however, only reviews works that aim to derive the shortest schedule length.

In [79], the authors propose a greedy algorithm to solve the data collection problem. In their algorithm, each packet has a priority that corresponds to the number of packets buffered at a node. To schedule packets to be transmitted in slot t, their algorithm first sorts all packets in decreasing priority order. Their algorithm then schedules as many packets in slot t as possible if doing so does not interfere with packets that are already scheduled in t.

In [26], the authors propose a link scheduling algorithm to minimize data collection time in a tree topology. They assume each node has at least one packet to be delivered to the gateway. Their proposed algorithm generates a schedule in a path-by-path manner. The key idea is to always schedule one packet from the sub-tree that has the most number of unscheduled packets destined for the gateway. Their proposed algorithm does not schedule packets to the same sub-tree consecutively. Instead, their algorithm first finds the shortest path from the source node of a packet to the gateway. Then their algorithm assigns an increasing positive integer

Work	**Centralized or Distributed**	**Transmission Mode**	**Objective**	**Solution**
Bao et al. [24]	Distributed	Multi-Transmit-Receive	Maximize capacity	Use a hash function to determine which links to activate
Steenstrup et al. [68]	Distributed	Multi-Transmit-Receive	Maximize capacity	Randomly pick a slot to send reservation request message
Sundaresan et al. [69]	Centralized	Multi-Transmit-Receive	Maximize capacity	Always color the node with the most neighbors
Chu et al. [70]	Centralized	Multi-Transmit-Receive	Maximize capacity	Assign slots to links based on link priority which is determined by delay and data importance
Chang et al. [71]	Centralized	Multi-Transmit-Receive	Maximize capacity	Gather non-interfering links into HPSs and combine HPSs into CPSs. Then assign slots to CPSs according to number of links
Hung et al. [72]	Centralized	Multi-Transmit-Receive	Minimize delay	Always schedules the link with the strictest delay requirement
Jawhar et al. [73]	Distributed	Multi-Transmit-Receive	Provide guaranteed end-to-end bandwidth	Source node sends reservation packets flows to reserve path and slots
Mumey et al. [74]	Centralized	Multi-Transmit-Receive	Minimize superframe length	Vertex coloring on the conflict graph. Use a previously used color on a vertex before picking a new color.
Deopura et al. [75]	Centralized	Multi-Transmit-Receive	Meet transmission demand of each link	Assign slots to links based on link demand
Bhatia et al. [76]	Centralized	Single-Transmit-Receive	Maximize capacity	Use LP to generate routing paths and a link schedule. Assigns links into the first non-interfering slot if the schedule generated by LP is not feasible
Wang et al. [77]	Centralized	Multi-Transmit-Receive	Maximize capacity	Use MILP to generate routing paths. Group the incoming links of each node into a link set and assign different slots to each link set.

Table 2.3: Comparison of works using smart antennas

to each link along the path from the source node, where the link assigned with integer k is activated in slot k; the number assigned to each link along a path is strictly increasing. In addition, a color k is assigned to a link if activating the link in slot k does not interfere with other links that are scheduled in slot k. Their algorithm stops when all packets are scheduled.

The authors in [80] aim to minimize data collection time. Each node has one packet to be delivered to a gateway node. They assume there are C channels and each node is able to use only one channel to transmit one packet in each slot. They also assume a spanning tree, rooted at the gateway node. They first formulate the link scheduling problem as an ILP. The decision variables are the links to be activated in slot t. They then generate the data collection schedule with the following two policies: *1)* node v_i, which is a child of the gateway, is scheduled to be a transmitter in slot t if it has the most number of buffered packets among unscheduled nodes, and v_i is not a transmitter in slot $t-1$, *2)* a node v_j that is not a child of the gateway is scheduled for transmission in slot t if its parent node is scheduled in slot $t-1$ and v_j has the most buffered packets among unscheduled nodes.

The authors in [81] study the fast data collection problem in duty-cycled WSNs. They assume a linear topology with a base station at one end of the topology. Each node is assigned a unique ID ranging from 1 to N based on the distance (in terms of hops) from the base station; node 1 and N are assumed to be respectively the closest and farthest from the base station. Each node is equipped with an omni-directional antenna that has a transmission range of one hop and interference range of two hops. They first propose a centralized algorithm that achieves the optimal data collection time; i.e., in the non-duty-cycled case. During slot t, the algorithm checks each node, starting with the node that has the most buffered packets. If two or more nodes have the same number of buffered packets, the algorithm picks the node nearest to the base station. After picking a node, the algorithm first checks whether the node is within the interference range of a transmitting node. If not, the node sends one buffered packet to its next hop neighbor to the BS. Once the base

station receives all packets, their algorithm stops. They then propose an algorithm for the duty-cycled case. Specifically, each slot is assigned one wake-up slot in a cycle of $T \geq 3$ slots. In each cycle, a node receives a packet in its wake-up slot and transmits in the following slot. To assign a wake-up slot to each node, their algorithm starts from node 1 to N, and assigns slot $t \leq T$ to each node. Initially, the value of t is equal to T. When a node is assigned a wake-up slot, t is reduced by one. If t equals zero, the value of t is reset to T.

The authors in [82] aim to minimize data collection time. They assume nodes are static and each node is equipped with an omni-directional antenna. The transmission and interference range of each node is one hop. Each node has one packet to be delivered to the sink node. They first propose a heuristic algorithm for data collection in linear topologies. The algorithm produces a schedule that is at least $3(N-2)$ slots in length, where N is the number of nodes. Their proposed algorithm generates a link schedule in a slot-by-slot manner. To generate the schedule for slot t, their algorithm goes through each node, starting with the node closest to the sink, and schedules as many non-interfering nodes in slot t as possible. Their algorithm terminates when all packets arrive at the sink node. The authors then extend their proposed algorithm to tree topologies. Their algorithm repeatedly picks the farthest un-scheduled node, and finds the shortest path from the node to the sink. Then they use the proposed heuristic algorithm on linear topologies to generate the schedule for all nodes along the path. The algorithm stops when all packets are scheduled.

In [83], the authors aim to minimize data collection time. Each node is equipped with an omni-directional antenna. They assume sufficient number of channels are available. Hence, interference is only caused by a node receiving more than one packet simultaneously. The central controller, located at the sink node, is aware of the network topology and the number of packets to be transmitted to the sink node. They assume a routing tree is given and rooted at the sink node. To generate the schedule for slot t, the controller first sorts all sub-trees, each rooted at one of the sink's child, in decreasing order according to the total number of packets in each

sub-tree. The controller then schedules one packet for node k, which is the k-th child of the sink if the sub-tree rooted at k has the most number of packets and k is not scheduled to transmit in slot $t-1$. Next, the controller goes through the sub-tree rooted at k and sets a node i as the transmitter in slot t if 1) node i has a packet to transmit, 2) d_i mod 2 is one, where d_i is the number of hops from i to the sink node, and 3) node i has the highest number of buffered packets amongst nodes on the same level of the sub-tree. Lastly, the controller goes through all nodes on other subtrees and sets a node j as the transmitter in slot t according to conditions 1) to 3).

In [84], the authors aim to minimize data collection time in wireless networks. Specifically, their algorithm generates a link schedule in a slot-by-slot manner. To generate the link schedule for slot t, their algorithm picks the sub-tree T_i, rooted at the i-th child of the gateway, that has the most packets to be transmitted to the gateway. Then their algorithm checks each node on sub-tree T_i, starting from the gateway's child. A node is scheduled to transmit in slot t if it has buffered packets and it does not interfere with transmissions already scheduled in slot t. Their algorithm stops when all packets are scheduled.

The authors in [85] consider generating the shortest data collection link schedule for tree topologies. A node is termed *eligible* if it has received all data packets from its descendants. A node is scheduled to transmit in slot t if it is an eligible node. Each *eligible* node is associated with a weight that is determined by the number of descendants. To generate the schedule for slot t, their proposed link scheduler called WIRES first sorts all *eligible* nodes in decreasing weight order. WIRES then checks each node in decreasing weight order and schedules an eligible node into slot t if it does not interfere with already scheduled nodes.

Table 2.4 summarizes solutions to the personalized broadcast and data collection scheduling problem. Only approach [26] considers directional antennas while other works assume omni-directional antennas. The authors in [80] and [83] assume there are multiple channels while other works assume nodes operate over a single channel.

Work	Antenna Type	Transmission Mode	Number of Channels	Solution
Bonifaci et al. [79]	Omni-directional	Single-transmit-receive	Single channel	Schedule nodes in decreasing weight order
Bermond et al. [26]	Directional	Single-transmit-receive	Single channel	Color each link using increasing integers along the path for a packet from the gateway to end node
Zhang et al. [80]	Omni-directional	Single-transmit-receive	Multiple channels	Schedule one packet for the node with the most buffered packets if the node is not scheduled in previous slot
Shen et al. [81]	Omni-directional	Single-transmit-receive	Single channel	Schedule one packet for the node with the most buffered packets
Choi et al. [82]	Omni-directional	Single-transmit-receive	Single channel	Checks each node starting with the one closest to the sink. Schedule as many non-interfering nodes in each slot
Zhao et al. [83]	Omni-directional	Single-transmit-receive	Multiple channels	Always schedule packets for a sub-tree that is not scheduled in previous time slot
Incel et al. [84]	Omni-directional	Single-transmit-receive	Single channel	Perform a BFS from the gateway and schedule links in BFS order
Malhotra et al. [85]	Omni-directional	Single-transmit-receive	Single channel	Schedule eligible nodes in decreasing weight order

Table 2.4: Comparison of recent personalized broadcast and data collection schedulers

Further, all these works assume each node is equipped with a single antenna. Thus, these approaches are not applicable when nodes have multiple antenna elements. In addition, these works do not consider nodes with MTR capability.

2.3 Forest Construction

As mentioned in Chapter 1, using multiple gateways further reduces the personalized broadcast schedule makespan. In addition, the packet delivery time is determined by the longest personalized broadcast makespan among all gateways. Thus, it is important to balance the load between gateways. This section reviews recent works that aim to generate a forest that consists of multiple routing trees; each of which

is rooted at a distinct gateway. An example WMN with two gateways G_1 and G_2 is shown in Figure 2.1 (a). Solid lines indicate links between nodes. Figure 2.1 (b) shows the corresponding routing forest. There are two trees in total, rooted at gateway G_1 and G_2, respectively.

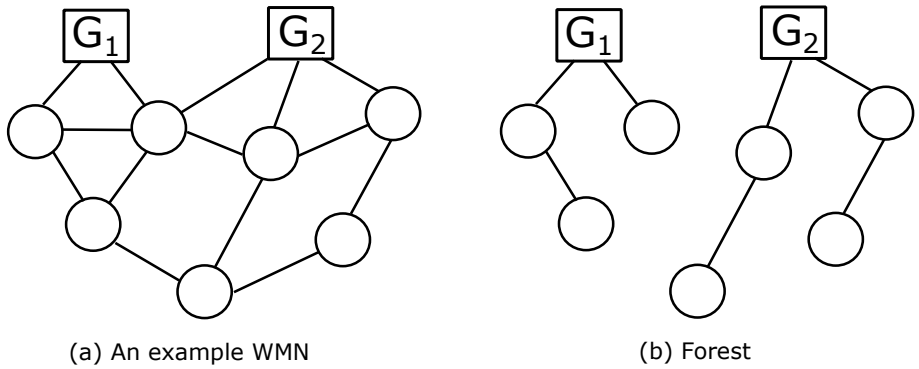


Figure 2.1: An example WMN and its corresponding routing forest

The authors in [86] propose a greedy load balancing routing algorithm, named GLBR, to generate a load balanced forest whereby links connected to each gateway carries near equivalent amount of traffic. In other words, all sub-trees in the forest have similar load. GLBR works in rounds. During each round, GLBR picks the sub-tree with the lowest load and connects one unconnected node to the sub-tree. Their algorithm terminates when all nodes are connected.

In [87], the authors aim to balance the load between different gateways by addressing the forest construction problem. Their proposed algorithm includes two phases. The first phase is used to group all nodes into different clusters, one to each gateway. During the second phase, their algorithm generates a tree for each cluster, where the generated tree is rooted at a gateway. Specifically, to group nodes into different clusters, the algorithm enumerates all possible cluster results/configurations. For a network with N nodes and K gateways, there are K^N different configurations from which their algorithm needs to find the best one. However, the best solution becomes intractable to compute as N and K become large. Thus, their algorithm removes 75% of the most imbalanced configurations. To pick these configurations, their algorithm calculates the difference in load between the heaviest and lightest

gateway. Then for each of the remaining configurations, their algorithm constructs a forest with K trees in round-by-round manner. In each round, their algorithm connects one node that is not on the tree to another node that is already on the tree. After constructing a forest for each remaining configuration, their algorithm calculates the T_{cycle} for each configuration; T_{cycle} is the total amount of time that each node takes to receive one packet. It then picks the one with the lowest value.

The authors in [88] aim to generate a load-balanced forest that maximizes network throughput. They first propose a forest construction algorithm to generate a routing forest. Their algorithm repeatedly connects a node to the tree with the lowest load. They then propose a load readjustment algorithm to balance the load between different gateways when the load changes. When the load of a gateway k exceeds a given load limit Q, their algorithm checks each node on the tree rooted at gateway k, starting from the node farthest from the gateway. Their algorithm swaps a node u and all its descendants from gateway k to another gateway j if i) node u has a neighbor that is on the tree rooted at j and ii) after swapping, the load of gateway j does not exceed Q. This swapping procedure continues until the load of gateway k is lower than Q.

The authors in [89] propose a forest construction algorithm to balance the load between different trees. They assume a network is organized as a forest with K trees; each rooted at one gateway. Their algorithm then balances the load between different trees by reconfiguring the topology. Each node keeps a table N containing its neighbors that are connected to a tree rooted at a different gateway. Each node also maintains a periodic timer. When the timer at node i, on the tree rooted at gateway k, expires, node i checks whether it is able to swap to another tree. To do this, node i checks all nodes in N, and picks a neighboring node j that is on the tree rooted at the gateway s with the lowest load. If the load of gateway s is lower than that of gateway k, their algorithm swaps node i from the tree rooted at gateway k to the tree rooted at gateway s.

The forest construction problem is similar to the load balance routing problem

with multiple gateways; see [90] [91] and [92]. The difference is that the resulting load balanced routing may not be a tree-based topology. In [90], the authors aim to balance the load between different gateways. Their proposed load balancing protocol has two phases: gateway discovery and load migration. In the gateway discovery phase, gateways broadcast an existence message. Upon receiving the message from gateway k, a node n decides to send/receive a message through gateway k if i) it has not received any existence message or ii) gateway k is nearer (in number of hops) as compared to its current gateway. During the load migration procedure, when the queue length at gateway k exceeds a given threshold, it picks the node that is generating the most traffic. The node is then sent a gateway change message. A node n that receives the gateway change message picks the next nearest gateway s and sends a change request message to s. The gateway change is successful if the load of gateway s is below the threshold after node n changes to gateway s. The authors in [91] propose a load balance routing protocol to balance gateway load. Each node knows one path to each gateway. To achieve this, each gateway broadcasts a root announcement (RANN) message into the network. After receiving a RANN message, a node records itself in the RANN message and broadcasts the message to its neighbors. A RANN message will be discarded if a node is already recorded in the received message. After a node receives an RANN message from a gateway, it knows one path to that gateway by checking the received RANN message. After knowing the path to all gateways, each node then picks the nearest gateway k (in number of hops) and replies with a route reply message (RREP) to inform gateway k that it will send/receive packets through it. To balance the load between gateways, the authors extend the approach proposed in [90]. Briefly, in [90], the proposed algorithm monitors the queue length at a gateway and migrates the node that generates the most traffic. However, in [91], their algorithm picks a migrating node based on traffic rate, packet loss ratio and the distance (number of hops) between the node and the gateway. The authors in [92] propose a load balancing routing algorithm that considers multiple gateways. Specifically, their

Work	**Final Topology**	**Metric**	**Solution**
Wen et al. [86]	Forest	Number of flows	Pick the sub-tree with the lowest number of flows and connect one node to the sub-tree
Das et al. [87]	Forest	Estimate amount of time that each node receives one packet	Enumerate all clustering results and discard the 75% most imbalanced ones. For each remaining cluster, repeatedly connect a node to its neighbor that is already connected to the forest
Xu et al. [88]	Forest	Volume of data received at each gateway	Repeatedly connect a node to the tree with the lowest load. Swap a node and all its descendants to another gateway if its current gateway exceeds a given load limit
Mhatre et al. [89]	Forest	Number of packets received at each gateway	Swap a node to another tree with a lower load.
Nandiraju et al. [90]	General	Queue length at gateway	Send a notification message to the node generating the most traffic. Node receives this message swap to another tree with a lower queue length
Maurina et al. [91]	General	Queue length at gateway	Send a notification message to the node with the highest weight, where the weight of a node is determined by traffic rate, packet loss ratio and the number of hops between the node and the gateway.
Yan et al. [92]	General	Bandwidth at gateway nodes	Find the shortest path to the nearest gateway with sufficient bandwidth.

Table 2.5: Comparison of recent forest construction and load balance routing schemes

algorithm goes through each node, starting with the node with the lowest node ID. For each node u, their algorithm finds the gateway k that is nearest to node u (in number of hops) and has a load below a given threshold. Then their algorithm finds the shortest path between u and k.

Table 2.5 compares the aforementioned forest construction and load balance routing approaches. Among these approaches, references [86], [87], [88] and [89] produce a forest while other approaches do not yield a tree-based topology, which is required by the personalized broadcast problem. All these approaches balance the load between different gateways using a metric; e.g., number of flows, total traffic, number of packets, queue length and delay. However, none of these approaches consider minimizing personalized broadcast makespan whilst balancing load.

2.4 Full-Duplex

As mentioned in Chapter 1, full-duplex communication is now viable. However, an efficient MAC protocol is required to exploit the full-duplex capability of nodes. Specifically, there are two types of full-duplex transmissions: bi-directional and relay. In bi-directional full-duplex transmissions, a node can transmit and receive to/from a neighbor simultaneously. However, in relayed full duplex transmissions, a node transmits to one node while receiving from another node at the same time. An example relay and bi-directional full-duplex transmission is shown in Figure 2.2, respectively. Solid arrows indicate packet transmissions. Figure 2.2 (a) shows full-duplex relaying being carried out by node B where it is able to receive from node A whilst simultaneously transmitting to node C. In Figure 2.2(b), node D and E employ bi-directional full-duplex mode to transmit and receive simultaneously. This section reviews recent MAC protocols proposed for full-duplex wireless networks. The fundamental problems addressed by these works include: channel contention, collision resolution and enabling full-duplex communications.

(a) Relay full-duplex (b) Bi-directional full-duplex

Figure 2.2: An example of relay and bi-directional full-duplex transmissions

In [17], the authors present and experimentally validate a full-duplex MAC protocol. Their MAC focuses only on the simplest case: exchanging data over a full-duplex link. Nodes use CSMA to contend for the channel. The transmission from the winning node is designated as the *primary* transmission. The receiver of the primary transmission starts to transmit its data to the primary transmitter immediately; this transmission is termed as the *secondary* transmission. In addition, if the time required by the primary and secondary transmission is different, a transmitter starts transmitting a busy tone if it has not finished receiving data frames. This ensures that the primary and secondary transmission stop at the same time.

The authors in [18] propose a full-duplex MAC protocol called FuMAC. It only considers bi-directional full-duplex transmissions. Nodes use CSMA for channel contention. The node that wins the channel initiates data transmission. The intended receiver transmits its data back to the sender to form a bi-directional transmission upon receiving the sender's data frame. They then propose a collision detection scheme. Specifically, each node is assigned a unique sequence, embedded in the header of its transmitted frame. When a full-duplex node hears a different embedded sequence other than its intended receiver, it knows there is another interfering transmitter nearby. A transmitter cancels its transmission if i) it hears a sequence that does not belong to the corresponding receiver or ii) it did not receive a packet from its intended receiver after initiating a transmission. Lastly, the authors propose a collision resolution scheme. Each node has a channel access probability p. Each node knows its own a collision intensity c, which is the average number of collisions that a node senses within a unit time. Nodes first calculate the optimal collision intensity c^* that results in the maximum capacity. During each time unit, if node n's collision intensity c is less than c^*, it increases its channel access probability p by a factor k, where $k > 1$. Otherwise, p is decreased by the same factor. This ensures the collision intensity of each node is around c^* and the capacity is maximized.

In [93], the authors propose a CSMA/CA based full-duplex MAC protocol. They assume each node is equipped with three antenna elements; two directional antennas for transmission and one omni-directional antenna for reception. They assume a linear topology and uni-directional data transmission. Their proposed MAC protocol differs from CSMA/CA as follows. First, they modify the half-duplex data transmission condition in CSMA/CA. Specifically, if a node detects the carrier and finds the destination of the on-going transmission is the node itself, the node is allowed to transmit. Second, their proposed MAC protocol removes the need to transmit an ACK. This is because data is transmitted one way only and an ACK message results in collision. Lastly, they remove the contention window. This is because they assume data is transmitted uni-directionally meaning no collision will be recorded

when multiple nodes transmit to a receiver.

The authors in [94] propose a MAC protocol for wireless multi-hop full-duplex networks. Their proposed MAC protocol, RFD-MAC, aims to enable both bi-directional full-duplex and relay full-duplex transmission. RFD-MAC adds a 1-bit information to each frame indicating whether the transmitter of the frame has successive frame to be transmitted. In addition, each node keeps a table to record the IP address of its neighboring nodes and whether a neighboring node has buffered frames. RFD-MAC works as follows. Nodes use CSMA to contend for the channel. The node that wins the contention transmits a frame; this is referred to as the *primary* transmission. To enable full-duplex transmission, the transmitter of the primary transmission then checks its table to pick a neighboring node as the transmitter of a *secondary* transmission. The node uses the following rules: 1) pick the receiver of the primary transmission if the receiver has a buffered frame or 2) pick a node that has a buffered frame destined to the transmitter of the primary transmission. Whenever a node finishes transmission before reception, it continues transmitting a busy tone until the reception ends.

In [95], the authors proposed a full-duplex MAC protocol called DAFD-MAC. It has a five-way handshake. Specifically, DAFD-MAC works as follows. Nodes use CSMA to contend for the channel. Successful nodes are called *primary* transmission nodes. They proceed to send a RTS frame to their intended receiver. The receiver of a RTS frame is referred to as the *secondary* transmission node. The secondary transmission node then checks its buffer and sends out an RCTS frame. This frame functions as the CTS frame of the primary transmission as well as the RTS frame of the secondary transmission; hence, a RCTS frame has two destinations. The two nodes that receive the RCTS frame then replies with a Set Network Allocation Vector (SNAV) frame. The primary and secondary transmission starts after the SNAV frame arrives at the destination node.

The authors in [96] propose a CSMA based full-duplex MAC protocol, called ContraFlow, that considers both relay and bi-directional full-duplex transmissions.

Each node performs CSMA to contend for the channel. The winning node, termed as the *primary* transmitter, starts to transmit its data frame immediately. The destination node is called the *primary* receiver. To take advantage of full-duplex communications, each node keeps a table that records, out of 10 transmissions, the number of packets received successfully by a neighbor. After receiving a data frame from a *primary* transmitter, the corresponding *primary* receiver picks a buffered packet to a destination node that has the highest successful transmission ratio. If the destination node is the *primary* transmitter, the *primary* receiver establishes bi-directional full-duplex transmission with the *primary* transmitter. Otherwise, the *primary* receiver initiates relay full-duplex transmission. If the *primary* receiver does not have any packets, it transmits a busy tone back to the *primary* transmitter to form a bi-directional transmission.

In [97], the authors propose a MAC protocol for multi-hop wireless networks. The protocol is based on CSMA and takes advantage of relay full-duplex transmission. They assume the path of each flow is given. The proposed protocol contains two phases: path establishment and data transmission. During the path establishment phase, a node, say A, that wins the channel sends a forwarding route request (FRR) packet to the next hop node of the head-of-line packet. FRR has the same functionality as the RTS packet in CSMA/CA. In addition, the FRR packet contains the address of all nodes on the path to the destination of the data packet. A node on the path, say B, that receives a FRR packet relays the FRR packet to the next hop node immediately. If a node, say B, did not hear the FRR packet relayed by the next hop node, say C, node B then assumes the FRR packet is not relayed successfully. The data packet will then stop at node B, which it will then try to forward onwards when it next wins the channel. The data transmission phase starts in parallel with the path establishment phase. When node A hears its FRR packet has been relayed by node B, it waits for t seconds before transmitting the data packet to B. Here, t is set to half of the time required to transmit the data packet. This is to ensure the transmission of the data packet does not interfere with node

C's reception of the FRR packet from B. Once node B starts receiving the data packet from node A, it waits for another t seconds before relaying the data packet to the next-hop node C in full-duplex mode.

The authors in [98] present a novel MAC protocol called Janus. The authors consider clients associated to an access point (AP); these clients do not exchange packets with each other directly. Janus eliminates collisions by scheduling interfering transmissions in different times. Janus works in rounds. At the start of each round, the AP sends a request packet to all clients requesting each client's buffered packet length and its interfering neighbors. Clients that receive the request packet reply to the AP in a pre-assigned order. The AP then generates a schedule as follows. It first calculates the time required for each of its incoming and outgoing packets using their length and channel condition. Next, it schedules both incoming and outgoing packets simultaneously for full-duplex transmissions while ensuring interfering transmissions do not overlap. To do this, the AP repeatedly checks whether its scheduled incoming transmissions is longer than its outgoing transmissions. If its incoming transmission is equal or shorter than its outgoing transmission, it picks an unscheduled incoming packet that requires the minimum time. Then the AP schedules the picked packet in the earliest available non-interfering time slot. On the contrary, if the AP's incoming transmission is longer than its outgoing transmission, it schedules one outgoing transmission to the earliest available time slot. Their algorithm stops when all packets are scheduled.

In [99], the authors proposed a CSMA based MAC protocol for full-duplex wireless networks. They assume the available bandwidth is divided into C orthogonal channels, where one of them is termed as the *primary* channel P_1 and is used for channel contention. Other channels P_c, where $2 \leq c \leq C$, are termed as *secondary* channels. Their proposed protocol works as follows. Nodes that have buffered data performs CSMA on channel P_1. The winning node, say A, starts transmitting its data frame through P_1 to destination node B. To ensure fairness, the amount of data that A can transmit through the primary channel after one successful channel

contention is bounded. If node A has more data than a given limit, it requests a *secondary* channel from the receiver to transmit additional data frames. This is achieved by setting the Channel Negotiation Bit (CNB) to one in the transmitted frame. Node B then informs node A the first non-interfering *secondary* channel P_c via the ACK for the data frame transmitted through P_1. Node A starts transmitting the additional data frame in channel P_c after receiving an ACK from B. To achieve full-duplex transmission, node B contends for the *primary* channel P_1 while receiving from A through P_c. Moreover, node A is able to receive from channel P_1 while transmitting to node B in a *secondary* channel to achieve full-duplex transmission.

To date, there are several MAC protocols that consider a full-duplex AP with half-duplex clients; e.g., [100] [101]. In these two works, only relay full duplex transmissions are supported because clients are only capable of half-duplex communications; i.e., they are unable to form bi-directional full-duplex transmission with the AP. In addition, both works consider the *capture effect* [102] on the client side. Specifically, a receiver is able to decode a desired packet even through it overhears multiple concurrent transmissions. However, to decode a packet successfully, transmissions must be asymmetric and the desired packet must have a higher signal strength [103]. Thus, a client is able to decode a packet transmitted by the AP successfully even though another client is transmitting to the AP simultaneously. In [100], the authors propose a full-duplex MAC protocol named A-Duplex. Specifically, clients and the AP use CSMA. Full-duplex will not be activated if the AP wins the channel. This is because once half-duplex clients sense a busy channel they will not start another transmission to the AP. If a client, say A, wins, it then sends an RTS packet to the AP. Once the AP finds it has a packet for another client B, the AP replies with an CTS to A and starts transmitting its data frame to B immediately. Upon receiving the CTS, client A checks whether the packet it transmits is shorter than the packet the AP transmits to B. If yes, node A delays its transmission to ensure both transmissions terminate at the same time. Otherwise, if the packet transmitted by A is longer, the AP transmits a busy tone after it finishes

transmitting packets to B.

In [101], the authors outline a TDMA solution, named FD^2 that operates in wireless networks with a full-duplex capable AP and half-duplex nodes. They assume the AP has K directional transmission antennas and K receive antennas while each client only has one omni-directional antenna. Specifically, FD^2 is divided into five phases. The first phase is used for information collection. To do this, the AP transmits a known sequence sequentially on each of its antennas. Clients record the received signal strength indicator (RSSI) of each transmitted sequence and then send the measured RSSI to the AP together with its queue information. In the second phase, the AP builds a conflict graph based on the collected RSSI. Each vertex in the conflict graph represents either an uplink queue (clients to AP) or a downlink queue (AP to clients). The third phase is used to generate the schedule. To do this, the AP first associates each queue with a priority that is set according to queue length and the waiting time of the head-of-line packet. The AP then goes through each queue in decreasing priority order and schedules one packet to be transmitted in slot t if it does not interfere with those that are already scheduled in t. The fourth and fifth phases are used for data transmission and acknowledgment, respectively.

Table 2.6 compares recent full-duplex MAC protocols. The authors in [17] [18] [98] [99] [100] and [101] only consider single hop transmissions, while other approaches consider multi-hop transmissions. However, in multi-hop approaches, references [93] and [97] only consider relay full-duplex transmissions while [94] [95] and [96] consider both bi-directional and relayed transmissions.

Work	Antenna Type	Channel Access	Relay and/or Bi-directional	Single or Multi hop	Innovations
Jain et al. [17]	Omni-directional	CSMA	Bi-directional only	Multi-hop	Real-world implementation. Use analog and digital self-interference cancellation to achieve full-duplex
Xie et al. [18]	Omni-directional	CSMA	Bi-directional only	Single hop	Real-world implementation on 802.11 MU-MIMO hardware. Use antenna elements to cancel self-interference
Miura et al [93]	Multiple directional for transmission and one omni-directional for reception	Modified CSMA	Relay only	Multi-hop	Use directional antenna to suppress interference.
Tamaki et al. [94]	Omni-directional	CSMA	Both	Multi-hop	Add a unique sequence in frame header for collision detection
Sugiyama et al. [95]	Directional	CSMA	Both	Single hop	Use directional antenna to suppress interference
Singh et al. [96]	Omni-directional	CSMA	Both	Multi-hop	Pick secondary transmissions according to successful transmission ratios
Askari et al. [97]	Omni-directional	CSMA	Relay only	Multi-hop	Delays data packet transmission to avoid collision between data and FRR packets
Kim et al. [98]	Omni-directional	TDMA	Both	Single hop	Schedule interfering transmissions in different time
Queiroz et al. [99]	Omni-directional	CSMA	Both	Single-hop	Divide available bandwidth into multiple orthogonal channels
Tang et al. [100]	Omni-directional	CSMA	Relay only	Single-hop	Real-world implementation with full-duplex AP and half-duplex clients
Aryafar et al. [101]	Multiple directional at AP and single omni-directional at clients	TDMA	Relay only	Single-hop	Schedule transmissions based on priority

Table 2.6: Comparison of recent full-duplex MAC protocols

2.5 Summary

This chapter reviews recent works that consider link scheduling in TDMA based WMNs with directional antennas, the personalized broadcast or data collection problem, load balance routing methods and link scheduling approaches for full-duplex networks. However, existing works leave the following gaps:

1. There are only a handful of distributed schedulers that target TDMA based WMNs with directional antennas [38] [40] [25], [58] [59] and [24]. References [38] and [40] use single directional antenna and thus they cannot be used in MTR WMNs. The approach in [25] uses a token-based method, where a node can only transmit when it has the token for all its links. As shown in Chapter 1, this degrades performance as it causes idle nodes. Consequently, it activates a sub-optimal number of links. The approach in [59] requires a rectangular topology and its application in general topologies remains an open question. References [58] and [24] use a randomized method, and consequently, in some time slots, network capacity is not maximized. Thus, Chapter 3 presents a distributed link scheduler that maximizes capacity in MTR WMNs with general topologies.

2. Personalized broadcast or data collection schedulers to date do not consider the MTR capability of nodes and their ability to send multiple packets to a neighbor. Moreover, no works that address the data collection or personalized broadcast schedule problem consider interference from neighboring transmissions or use antennas to cancel interference. Thus, applying these works in MTR-WMNs is likely to yield a schedule with a large makespan, which in turns causes nodes to have large delays. To this end, Chapter 4 presents a novel scheduler that minimizes the personalized broadcast schedule in MTR-WMNs. Apart from that, forest construction approaches and load balance routing methods reviewed in Section 2.3 do not consider MTR WMNs. Further, these approaches do not consider deriving a personalized broadcast

schedule. Consequently, Chapter 4 also contains a heuristic algorithm that constructs a balanced forest in MTR-WMNs that allows distinct packets to be delivered to nodes in minimum time.

3. Most full-duplex approaches to date are based on CSMA. Consequently, there is an opportunity to develop TDMA based approaches; e.g. [98] and [101]. However, both [98] and [101] only consider single hop packet transmissions. In addition, [101] only considers a full-duplex AP while clients work in half-duplex mode. Further, no works consider delivering both uplink and downlink packets from/to clients over multiple hops in minimum time. Moreover, in CSMA approaches, even though nodes are capable of full-duplex communications, they may not have buffered packets to transmit. Consequently, their throughput is low. Thus, Chapter 5 focuses on generating the shortest TDMA superframe schedule to deliver both uplink and downlink packets in full-duplex MTR WMNs.

Chapter 3

A Distributed Maximal Link Scheduler for Multi Tx/Rx Wireless Mesh Networks

As shown in Chapter 2, there are a number of works that address the link scheduling problem for TDMA based WMNs with directional antenna. However, there are only a few distributed schedulers that aim to maximize capacity in MTR WMNs. Recall that a key novelty of MTR WMNs is that nodes are able to transmit to or receive from multiple neighbors simultaneously. Thus, a key challenge is to design a scheduler that utilizes the MTR capability of nodes to maximize network capacity.

Henceforth, this chapter presents a distributed link scheduler that generates a superframe with minimal length while maximizing the number of concurrent links in MTR WMNs. Specifically, the proposed algorithm, Algo-d, maximizes concurrent links in each time slot by solving the MAX-CUT problem in a distributed manner. Specifically, in each time slot t, Algo-d divides all nodes into two maximally connected sets, where the number of unscheduled links across the two sets are maximized. Links across the two sets are scheduled in slot t. The results, see Section 3.5, show that Algo-d outperforms the state-of-the-art distributed schedulers [25] [24]. Specifically, for a network with 70 nodes, Algo-d schedules 28% and 270%

more links in each slot and derives a superframe that is 3.5 times shorter on average as compared to JazzyMac and ROMA, respectively.

The rest of this chapter has the following structure. Section 3.1 presents the network model. A description of the problem is shown in Section 3.2. Solutions are outlined in Section 3.3. Section 3.4 includes the characteristics of Algo-d and experimental results are shown in Section 3.5. Section 3.6 concludes the chapter.

3.1 Preliminaries

Consider a MTR WMN modeled as a directed graph $G(V, E)$, where V denotes the set of *static* vertices/nodes/routers and E denotes the set of directional links. Let $v_i \in V$ denote a node i with k_i radios; each radio is connected to a directional antenna and is used to communicate with a neighbor. Let $e_{ij} \in E$ represent a directional link from node i to j. Denote N_i to be the set of node i's neighbor. A key constraint, however, is that nodes are not allowed to transmit *and* receive concurrently. From here on, this constraint is defined as the no Tx-Rx constraint; see Figure 3.1. Assume each TDMA superframe S contains $|S|$ time slots and each slot is sufficient for a data packet and an ACK. Time synchronization is critical in both centralized and distributed solutions. Assume that nodes are synchronized and time is discretized into slots. This is reasonable because nodes are static, and current clock synchronization solutions such as [104] can achieve a precision of 1.5 microseconds in single hop scenarios and an average precision of 0.5 microseconds per hop for multi-hop cases. Note that the clock drift at each node will accumulate in multi-hop scenarios. An alternative solution is to use a GPS module synchronization. The schedule in each slot t can be represented as a $|E|$ dimensional activation vector $\mathbf{e^t}$, where each element corresponds to a link (i, j) and is set to one if the link is active in slot t. In particular, the vector $\mathbf{e^t}$ denotes the set of links that adhere to the *no-Tx-Rx* constraint. Note, a transmission set $\mathbf{e^t}$ is said to be *maximal* if no other links can be added into it without violating the *no-Tx-Rx* constraint. Let $\mathcal{B}$ be the

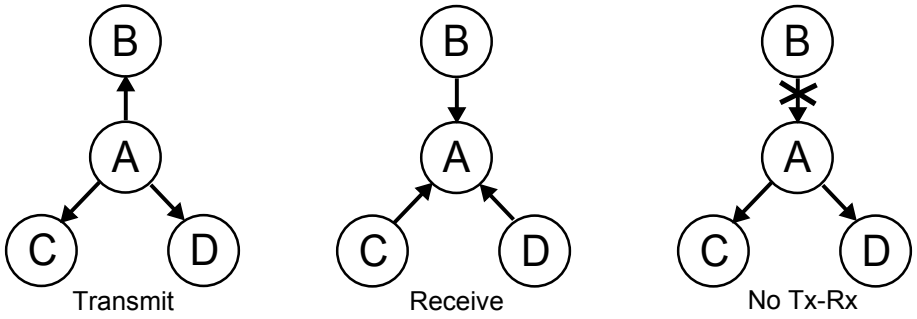


Figure 3.1: An example of the *no-Tx-Rx* constraint

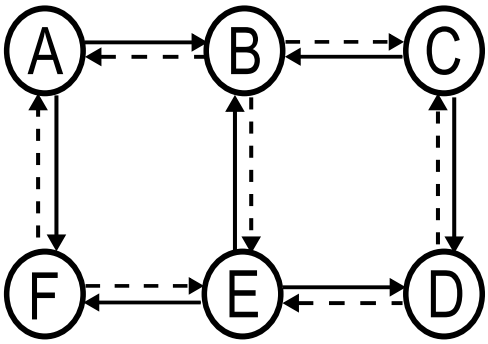
set containing all maximal feasible transmission sets. Also define λ_t to be a binary variable, i.e., $\lambda_t \in \{0, 1\}$, that indicates whether transmission set $\mathbf{e^t}$ is included in the superframe S. Also $\mathbf{1}$ is a $|E|$ dimensional vector containing all 1s.

Notation	**Definition**
V	The set of vertices/nodes/routers
v_i	Node i
E	The set of directional links
e_{ij}	Directional link from v_i to v_j
$\mathbf{e^t}$	Set of links that can transmit concurrently in slot t
N_i	Neighbors of node i
S	TDMA superframe
$\|S\|$	Superframe length, number of slots in the superframe
(x_i, y_i)	The variable x_i denotes the total number of nodes in S_1 that v_i points to, and y_i denotes the total number of nodes in S_2 that v_i connects to.
Δ_i	The difference between x_i and y_i, i.e., x_i - y_i.
$\mathcal{B}$	The set containing all maximal feasible transmission sets.
λ_t	A binary variable that indicates whether transmission set $\mathbf{e^t}$ is included in the superframe S.

Table 3.1: Notations and definitions

Let S_1 and S_2 be two maximally disjoint connected sets. Initially all nodes are in S_1, and S_2 is an empty set. A node i can either be in S_1 or S_2 in each transmission schedule. That is, for a given slot t, S_2 contains all nodes that transmit in slot t, and S_1 contains all nodes that receive in slot t. As an example, in Figure 3.2, for slot 1, node B, D, F are in S_2 and node A, C, E are in S_1.

Each node v_i is associated with two variables: x_i and y_i. The former denotes the total number of nodes in S_1 with a link to v_i. On the other hand, the variable y_i denotes the total number of nodes in S_2 that links to node v_i. As an example, consider the network shown in Figure 3.2. Initially all nodes are in S_1.



Superframe	
slot-1	slot-2
B→A	A→B
B→C	C→B
B→E	E→B
D→C	C→D
D→E	E→D
F→E	E→F
F→A	A→F

Figure 3.2: An example WMN topology and its corresponding schedule

The (x_i, y_i) value of each node is $(2,0), (3,0), (2,0), (2,0), (3,0), (2,0)$ for nodes A to F respectively. After slot-1, the (x_i, y_i) value for nodes A to F becomes $(2,0), (0,3), (2,0), (0,2), (3,0), (0,2)$, respectively. Lastly, define Δ_i to be the difference between x_i and y_i; i.e., $\Delta_i = x_i - y_i$. Table 3.1 gives a summary of the notations used throughout the remainder of this chapter.

3.2 The Problem

The link scheduling problem is to maximize the number of links activated in each time slot and to compute a minimal superframe length – both of which increase network capacity. The problem at hand can be formulated as follows:

$$\textbf{min} \sum_{k=1}^{|\mathcal{B}|} \lambda_k \tag{3.1}$$

subject to:

$$\sum_{t=1}^{|\mathcal{B}|} \lambda_t e^t \geq \mathbf{1} \tag{3.2}$$

In this thesis, the designed algorithm heuristically minimizes the superframe

length by maximizing the number of links activated in each slot. Specifically, Algo-d generate a MAX-CUT for each transmission set $\mathbf{e^k}$ so that each set has the maximal number of active links. Consider the set $\mathcal{B}$ with three members: $\{(1,0,1),(0,0,1),(0,1,0)\}$. Recall that each set constitutes the set of links that can be activated whilst adhering to the no transmit *and* receive constraint. The objective is to pick a combination that meets the following constraint: all links are activated at least once. In this example, we thus have the objective $min(\lambda_1+\lambda_2+\lambda_3)$ subject to $\lambda_1 e^1 + \lambda_2 e^2 + \lambda_3 e^3 \geq 1$ where e^1, e^2, e^3 correspond to sets $(1,0,1),(0,0,1)$ and $(0,1,0)$ respectively. Here, selecting sets $(1,0,1)$ and $(0,1,0)$ suffice and yields an objective function value of two.

In the said problem, each transmission set constitutes a MAX-CUT. As shown in [105], the best centralized solution using semi-definite programming is able to achieve an approximation ratio of 0.878. This thesis, however, considers computing the optimal family of transmission sets in a *distributed* manner. In other words, the goal is to design a distributed algorithm to solve the MAX-CUT problem to yield the minimal number of transmission sets, or equivalently the shortest superframe length, as reflected in Equation (1), that affords each link at least one activation slot, (see Equation (2)) using only one hop neighbor information. To date, no distributed algorithm for MAX-CUT exists; Algo-d is the first one. Note, in this thesis, link load and queue length are not considered, and the development of a suitable scheduler considering link load and queue length is deferred to be a future work.

3.3 Solution

The proposed solution is based on Algo-2, a centralized scheduler; see Chapter 2 and [31] for details. Specifically, Algo-d, divides the topology into two maximally connected sets S_1 and S_2, but in a distributed manner. The key idea is to determine and update the x and y values of each node using only local information. More specifically, the x and y values help determine the set membership of nodes that

yield the maximal cut. As will be shown later, the node i with the maximum and positive Δ_i value amongst its 1-hop neighbors broadcasts a message to all its neighbors informing them that it will move to S_2 to become a transmitter. Then all nodes update their (x, y) values. A node concludes it has the schedule, i.e., e^t, for a slot t when the Δ value of all one-hop neighbors is equal to or less than zero.

The next section first outlines a key function used by Algo-d to transmit messages. Details of Algo-d is presented in Section 3.3.2, followed by an optimization to increase link activation in each slot. After that, Section 3.3.4 presents a protocol, called *RootRoute*, to inform nodes the start of the superframe.

3.3.1 Channel Access

Initially, Algo-d makes use of a random channel access function. This is required as nodes do not yet have an assigned slot. This function, labeled $TRANSMIT(msg, t_s)$, is used to transmit messages listed in Table 3.2 in a given time slot t_s. The function works as follows. Assume at time slot t, node i needs to send information to its neighbors. It sets itself to transmit in a slot with probability $\frac{1}{kW}$. Here k is a constant integer multiplier, and W is a network wide fixed constant, meaning nodes are not required to learn W. Clearly, selecting larger values for kW will reduce the probability of collision. If a collision occurs, i.e., there is no acknowledgment, node i retransmits for a maximum of Φ times. Note, a collision only occurs when neighboring nodes experience concurrent transmit *and* receive simultaneously. This is reasonable as directional antennas have high gains and any links with low SNR can be removed before scheduling. Note that unlike the omni-directional case, a node is allowed to receive from multiple neighbors simultaneously. In the experiments, described in Section 3.4 and Section 3.5, k is set to 3 and W to the half of maximum node degree to reduce the probability of conflicts. Also, Φ is set to 3.

Message	Description
Txy	Used to inform all neighbors a sender's (x_i, y_i) value.
InSet2	Used to inform all neighbors that the sender is moving to S2.
Updxy	Used to inform all neighbors that the sender's updated (x_i, y_i) value.
SchComp	Used to inform all neighbors the generated schedule.
FinishSched	Used by a node to inform its parent(s) that it has entered the ScheduleComplete state and also its slot number.
StartSFSlot	Sent by the root node to mark the start of the superframe
IRoot	Used to discover and construct a route to the node with the lowest ID

Table 3.2: Description of messages

State	Description
BootUp	Initial state
TxXY	Tuple $[i, 1, BootUp, t_s]$ has been initialized. Ready to exchange (x, y) with neighbors
WaitXY	Have sent (x, y) to all neighbor; wait to receive (x, y) or Δ from neighbor.
TxInset2	Ready to send InSet2 to all neighbors
ScheduleEnd	Wait for other nodes to finish their schedule for the current slot
ScheduleComplete	Wait for other nodes to activate all their links
RootRoute	Used to determine the node with the lowest ID
WaitSFrame	A node awaiting the start of the superframe

Table 3.3: Description of each state

3.3.2 Link Scheduling

This section explains how Algo-d determines a link schedule. The goal is to minimize the superframe length by maximizing the number of links activated in each time slot; i.e., generate a MAX-CUT for each slot. Algo-d divides all nodes into two maximally connected sets, in a distributed manner. A node can be in any of these eight states: **BootUp**, **TxXY**, **WaitXY**, **TxInset2**, **ScheduleEnd**, **ScheduleComplete**, **RootRoute** and **WaitSFrame**. Figure 3.3 shows the corresponding state diagram. Each node maintains the following tuple: $[ID, Set, State, Slot]$. Each node has a unique ID. The *Set* variable stores the set which a node belongs to, i.e., Set=1 if it is in S_1 or Set=2 if in S_2. The variable *State* stores a node's current state, and *Slot* represents the time slot it is deriving a link schedule for. Table 3.2 shows the list of messages exchanged by nodes and a short description of each state is shown in Table 3.3.

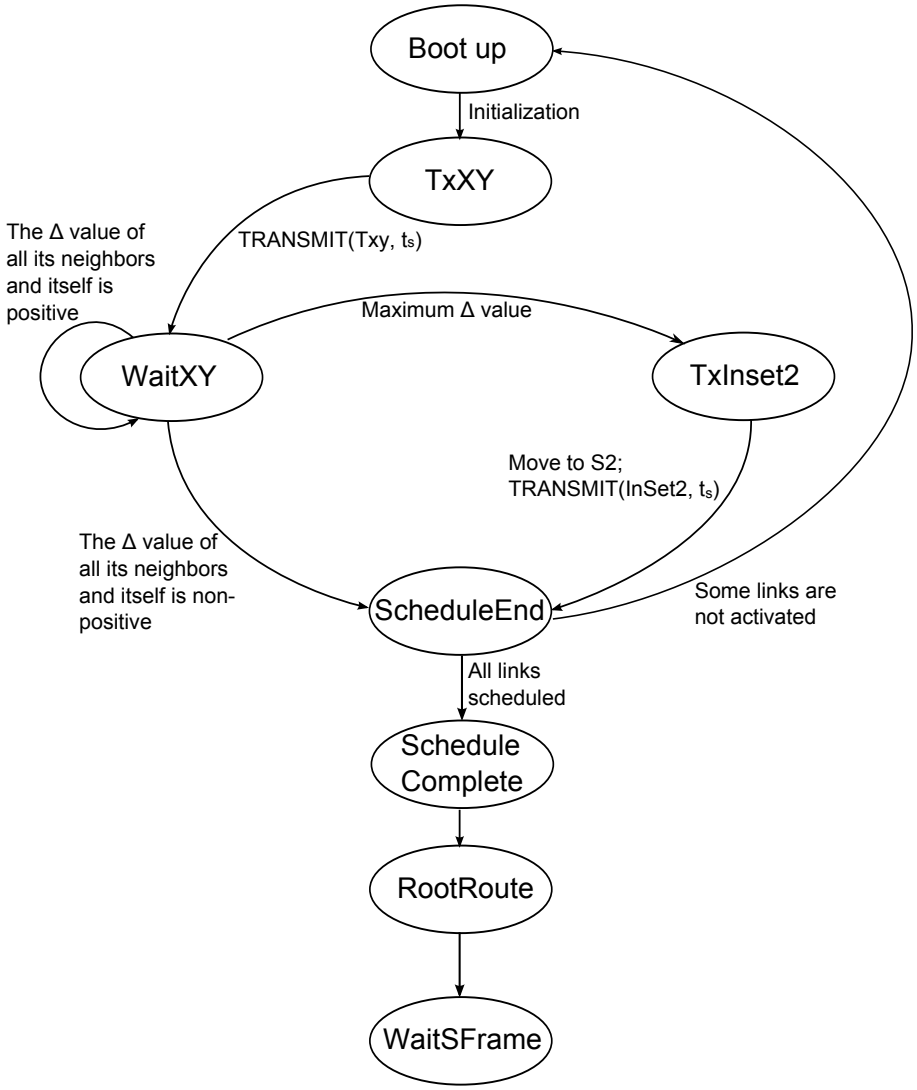


Figure 3.3: State diagram of Algo-d

The main idea is to use the Δ value of nodes to determine set membership; i.e., S_1 or S_2. Consider node i. To generate the link schedule for slot t_s, it starts from the **BootUp** state, and creates the tuple $[i, 1, BootUp, t_s]$. After initialization, it moves to the **TxXY** state. It then transmits its (x, y) value to all its neighbors; i.e., by calling $TRANSMIT(Txy, t_s)$. After that, it moves to the **WaitXY** state. Node i then checks if it has received all its neighbors' (x, y) value. If not, node i will wait until reaching a timeout value of $3|N_m|kW$ for a neighbor node m whose (x, y) value it has not received. Here $|N_m|$ is the node degree of m, which can be set to W by default. Consequently, as different neighbors have a different node degree, node i has a different timeout value for each neighbor. If it has, node i will then check if its

Δ value and all its neighbors' Δ value is equal to or less than zero. If both are true, then node i moves to the **ScheduleEnd** state. On the other hand, if node i's Δ value is positive, it will check if its Δ value is maximum as compared to its one-hop neighbors. If it is, it then checks whether it has the lowest ID among its neighbors that have the same Δ value. If there are no neighbors with the same Δ value or node i has the lowest ID, it will move to the **TxInset2** state. Otherwise, node i will stay in the **WaitXY** state. If node i is in the **TxInset2** state, it firstly moves itself to S_2, meaning it will transmit in slot t_s. Then node i sends an InSet2 message to all its neighbors to inform them that it will be in S_2. After that node i's neighbors that received the InSet2 message update their Δ value and inform their neighbors via $TRANSMIT(Updxy, t_s)$. Note, InSet2 and Updxy messages are broadcasted directionally to all neighbors. Nodes that are already in S_2 will ignore Inset2 and Updxy messages.

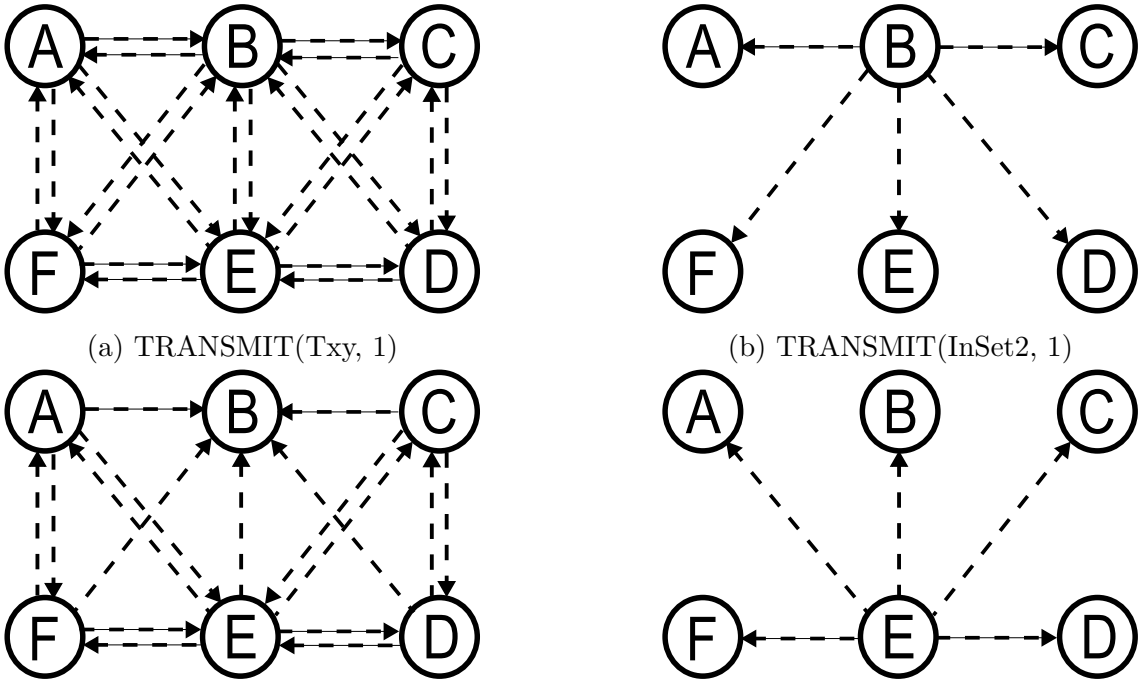
There is a small probability that packets are not delivered to a receiver node after Φ times. In this case, a sender will mark the link to the receiver and itself as 'scheduled' for the current round, and tries to schedule the link again in the next round. Similarly, if a node does not receive the (x, y) value of a neighbor m for $3N_mkW$ slots, it will also mark the link as 'scheduled' in the current round.

When all nodes enter the **ScheduleEnd** state, they have the link schedule for slot t_s. That is, in slot t_s, nodes in S_2 transmit, and those in S_1 receive. Each node checks if all its links have been scheduled. If not, it moves to the **BootUp** state to generate the link schedule for slot $t_s + 1$. Otherwise, it moves to the **ScheduleComplete** state. Note that when generating the link schedule for slot t_s+1, each node does not consider previously activated links while calculating the (x, y) value. Upon entering this stage, the process mentioned in Section 3.3.4 is initiated and the node enters the **WaitSFrame** state to await the start of the superframe.

The steps that Algo-d uses to determine the schedule for the 'two-boxes' topology shown in Fig 3.5 are as follows. Initially, all nodes are in the **BootUp** state and each of them sets a tuple $[i, 1, BootUp, 1]$. Then node i transmits its (x_i, y_i) value

to its neighbors, i.e., $TRANSMIT(Txy = (x_i, y_i), 1)$, and moves to the **WaitXY** state, see Figure 3.4a. The (x_i, y_i) value of each node is $A(3, 0)$, $B(5, 0)$, $C(3, 0)$, $D(3, 0)$, $E(5, 0)$ and $F(3, 0)$. After each node receives all its neighbors' (x_i, y_i) value, it finds all of them to be positive. Each node then checks if it has the maximum Δ_i value among its neighbors. Node B realizes that it has the maximum Δ_i value and lowest ID among its neighbors. It thus moves to S_2 and sends the message $TRANSMIT(InSet2, 1)$ to all its neighbors, and moves to the **ScheduleEnd** state; see Figure 3.4b and Figure 3.3 respectively. As nodes A, C, D and F do not have a maximum Δ_i value as compared to their neighbors, they remain in the **WaitXY** state. Although node E knows it has the maximum Δ_i value, its ID is not the lowest, i.e., its larger than B. So it also stays in the **WaitXY** state. Nodes that receive the $InSet2$ message then update their (x_i, y_i) value; i.e., we have $A(2, 1)$, $C(2, 1)$, $D(2, 1)$, $E(4, 1)$ and $F(2, 1)$. Each node then informs its neighbors of its new (x_i, y_i) value using $TRANSMIT(Updxy, 1)$, see Figure 3.4c. As node B is already in S_2, it will ignore this message. Node E then realizes that it has the largest and positive Δ_i value and moves to S_2 and use $TRANSMIT(InSet2, 1)$ to inform all its neighbors, see Figure 3.4d. Again, as node B is already in S_2, it will ignore this message. Each of node E's neighbors then updates its own (x_i, y_i) value. As all nodes observe the Δ_i value of themselves and their respective neighbors to be less than zero, the schedule for slot 1 is determined. That is, nodes in $S_2 = \{B, E\}$ will transmit in slot-1 while those in $S_1 = \{A, C, D, F\}$ receive. After that, each node removes the links activated in slot-1; see Figure 3.5b. Nodes that have un-activated links move to the **BootUp** state and restart the process to generate a link schedule for slot-2; for the example, all nodes move to the **BootUp** state. To aid understanding, the (x_i, y_i) value of each node for the first slot is outlined in Table 3.4.

Each node then updates its tuple as $[i, 1, BootUp, t_s]$ and repeats the same procedure to generate the schedule for slot 2, 3 and 4; see Figure 3.5b, Figure 3.5c and 3.5d respectively. This procedure repeats until all nodes enter the **Schedule-Complete** state. As a result, in slot 1, nodes B and E transmit and all other node



(a) TRANSMIT(Txy, 1)

(b) TRANSMIT(InSet2, 1)

(c) TRANSMIT(Updxy, 1)

(d) TRANSMIT(InSet2, 1)

Figure 3.4: Message exchange in the first slot

receives. In slot 2, nodes A, C, D and F transmit and nodes B, E receive. In slot 3, nodes A, B, and C transmits and other node receives. In slot 4, nodes D, E and F transmits. The final superframe length and links activated in each slot is exactly the same as Algo-2. Thus in this topology, Algo-d has the same performance as Algo-2, its centralized counterpart, in terms of superframe length as well as network capacity.

Message	**A**	**B**	**C**	**D**	**E**	**F**
After TRANSMIT(Txy, 1)	**(3,0)**	**(5,0)**	**(3,0)**	**(3,0)**	**(5,0)**	**(3,0)**
After node B TRANSMIT(InSet2, 1)	**(3,0)**	–	**(3,0)**	**(3,0)**	**(5,0)**	**(3,0)**
After each node receives B's InSet2 message and TRANSMIT(Updxy, 1)	**(2,1)**	–	**(2,1)**	**(2,1)**	**(4,1)**	**(2,1)**
After node E's TRANSMIT(InSet2, 1)	**(2,1)**	–	**(2,1)**	**(2,1)**	–	**(2,1)**
After each node receives E's InSet2 message and TRANSMIT(Updxy, 1)	**(1,2)**	–	**(1,2)**	**(1,2)**	–	**(1,2)**

Table 3.4: The (x, y) value of each node in the first slot

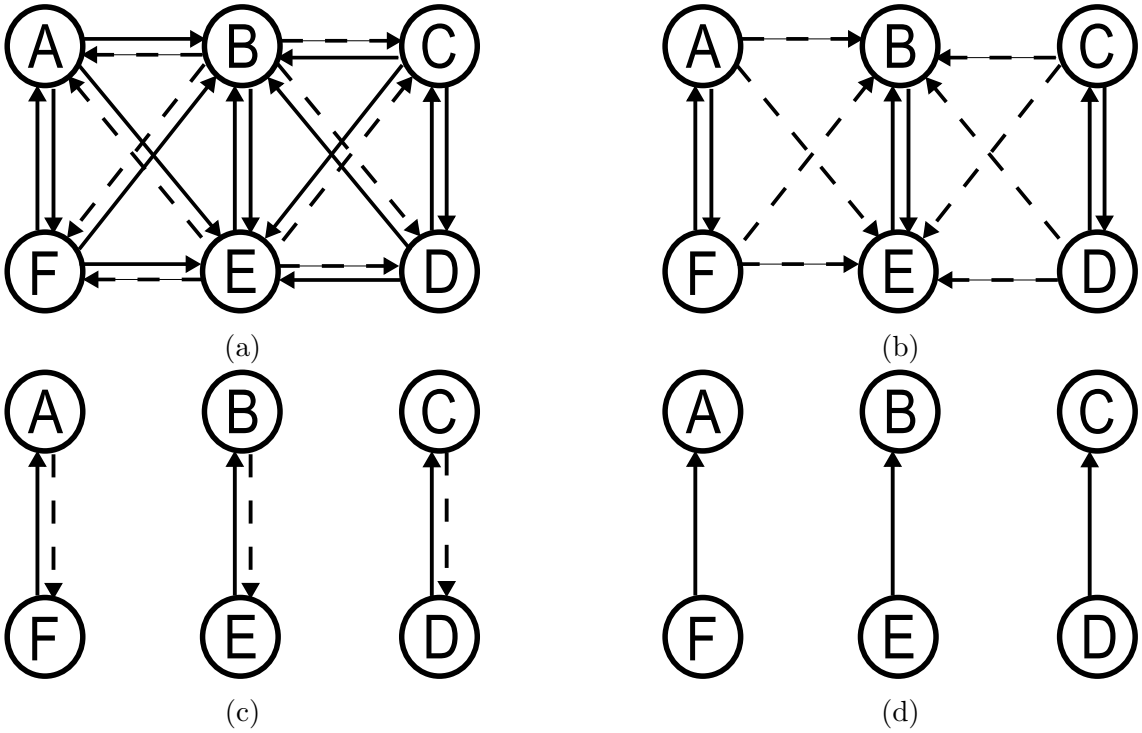


Figure 3.5: Two-boxes topology

3.3.3 Opportunistic Links

This section outlines how Algo-d adds more links in each slot. This optimization is required because the fact exists that there are more links scheduled in earlier slots as compared to those in latter slots. For example, in Figure 3.5, in the first slot, Algo-d schedules eight links. However, in the last slot, Algo-d only schedules three links. In fact, experimental results show that in the last slot, Algo-d schedules on average only $\frac{1}{10}$ of the links in the first slot. Remark that this behavior is correct because initially there are more un-scheduled links and as Algo-d picks links based on the highest Δ value, inevitably there will be fewer links later on. These observations mean that there are opportunities to improve network capacity in each slot by adding already scheduled, so called opportunistic, links. Formally, we have the following definitions.

Definition 1. Consider a link (i, j) that is activated for the first time in slot p. Each link (i, j) activated in any subsequent slot $r > p$ is called an opportunistic link or o-link.

Definition 2. A node i is an upgradeable node or *u-node* if it is in S_1 and all its

neighbors are in S_1.

Note that opportunistic links do not increase the superframe length. Further, opportunistic links are added after dividing nodes into S_1 and S_2 as described in Section 3.3.2. A key requirement when adding these links is that they must not violate the no-Tx-Rx constraint. Opportunistic links are generated in two steps: (i) u-node with the lowest ID among neighbors in the same set moves to S_2, and (ii) each node in S_2 activates all outgoing links to nodes in S_1. Note that more opportunistic links can be generated if u-node with the maximum number of neighbors in S_1 is selected to be moved to S_2. However, such an approach will require additional message transmissions among the u-nodes, adding more complexity to Algo-d, and hence is not recommended. Step (ii) is straightforward since Algo-d allows each node in S_2 to activate all outgoing links to its neighbors in S_1.

The optimization procedure works as follows. Consider the network shown in Figure 3.6a. Dotted links indicate those scheduled previously and four solid arrows are the links that will be activated in the current slot, i.e., non o-links. We see that $S_2 = \{B, D, F\}$, $S1 = \{A, C, E, G, H, I\}$, and H is the only u-node; thus node H moves to S_2. In Step (ii), each node in $S_2 = \{B, D, F, H\}$ activates its links to its neighbors in $S_1 = \{A, C, E, G, I\}$. Figure 3.6b shows the final schedule with twelve link activations, eight of which are o-links, e.g., e_{BE} and e_{HG}. Numerical results in Section 3.5 show that Algo-d improves network capacity by a further 119% due to this optimization.

3.3.4 Superframe Start Time

A fundamental problem is determining when nodes start using their allocated schedule. This is addressed by constructing a route to the node with the lowest ID. This root node upon finding all its children have entered the **ScheduleComplete** state then sends out a StartSFSlot message to all nodes. This message contains the start time $t_f = t + 2\mathcal{D}$, where $\mathcal{D}$ is the network diameter and t is the current time. Note

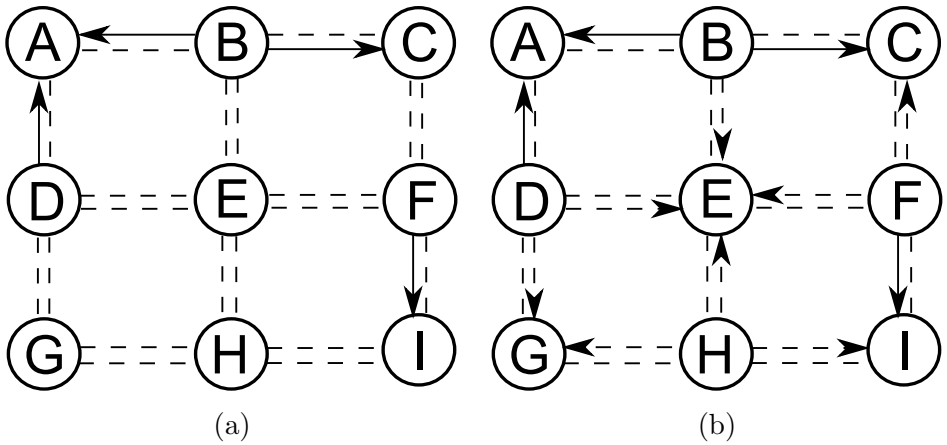


Figure 3.6: Improving a schedule with opportunistic links. Solid arrows are scheduled links and dotted arrows are opportunistic links

that the root can determine the superframe length from the maximum slot number contained in IRoot messages, explained later.

Assume each node i maintains (i) a set P_i containing neighbors marked as *parents* and is initially set to null, (ii) a variable tID^i that is initially set to a node's ID, (iii) a function $N_2^i(ID_x)$ that returns true if ID_x is higher than all its two hops neighbors' ID, including its own ID, and (iv) a set called ζ_i, which contains all neighbors marked as *children*. It is worth noting that all messages are sent using the TRANSMIT(.) function. Also note that two hop information can be readily obtained via HELLO messages because they contain a node's ID as well as that of its neighbors; i.e., from a HELLO message, a node is able to determine its neighbor's ID and the ID of said neighbor's neighbors.

A node with the lowest ID, say r, amongst its two-hop neighbors sends out an IRoot message to all its neighbors, so called children, and adds them into the set ζ_r. The IRoot message contains node r's ID, denoted as M_{ID}. Now consider a node, say i, that receives an IRoot message from neighbor p. If $N_2^i(M_{ID})$ and $M_{ID} > tID^i$ are true, the message is discarded because node i knows of at least one node with a lower ID. If $M_{ID} = tID^i$ and p is not in P_i, it is added to P_i; this means node i has more than one parent that leads to the root node. If $M_{ID} < tID^i$, node i sets $tID^i = M_{ID}$, $P_i = \{p\}$ and $\zeta_i = \emptyset$. It then forwards the IRoot message to all

neighbors except to those in P_i. Moreover, node i adds these neighbors into its set ζ_i. If node i finds that it has no children, i.e., all neighbors are in the set P_i, it is a leaf node.

Upon entering the **ScheduleComplete** state, a node i sends a FinishSched message to all its parents if (i) it is a leaf node, or (ii) it has received a FinishSched message from *all* its children. It then moves into the **WaitSFrame** state. Note, each node also includes its slot number and those of its children in the FinishSched message. Once the root receives a FinishSched message from all its children, it notes the allocated slots and sends a StartSFSlot message to them. In turn, they propagate the StartSFSlot message to their children and so forth. Upon receiving a StartSFSlot message, a node records the superframe length, marks the start of the superframe and prepares to transmit in its allocated slot.

Note that the procedure to find the root node can be enhanced to have nodes resend IRoot and FinishSched messages after a given timeout. That is, a node can resend its FinishSched message after waiting for a StartSFSlot message for some period of time. Similarly, a candidate root node can send out an IRoot message if none of its children have responded with a FinishSched message or it has not received another IRoot message with a lower ID for a given period of time.

3.4 Analysis

This section outlines several properties of Algo-d on simple as well as general topologies. This section also analyzes the complexity of Algo-d.

Proposition 1. *For a node i, the probability of a successful transmission within Φ attempts is $(1-P_s)^{\Phi-1}P_s$, where P_s is the probability of a successful transmission in each slot.*

Proof. First, a node i picks a slot out of kW slots with equal probability. If N contenders are present then $N-1$ nodes will all need to choose a slot other than

the one chosen by i; hence, the probability of a success transmission in each slot is $P_s = (1 - \frac{1}{kW})^{N-1}$. However, a node could experience $\Phi - 1$ failures before a success. Hence, the probability of success on the Φ-th try is $P(\Phi) = (1 - P_s)^{\Phi-1} P_s$. □

Note that $(1 - P_s)^{\Phi-1} P_s$ is an upper bound as N reduces over time whenever they are successful in their transmission.

Lemma 1. *The Δ value of each node is guaranteed to reduce to less than or equal to zero.*

Proof. Consider a node v. Suppose it has the highest Δ value amongst neighboring nodes. Then it will move itself to S_2, meaning its Δ value is zero. Recall that if it has equal Δ value with a neighbor, then the node with the lower ID wins, meaning if node v has the lowest ID, then its Δ value will be zero. Otherwise, its neighbor will enter S_2, meaning node v's Δ value will reduce by at least one.

Now consider the case where node v deems a neighbor to have a higher Δ value. The neighbor could either move into S_2 and thereby proving the lemma or the said neighbor's neighbor, say z, has a higher Δ value. However, node z may have a neighbor with a higher Δ value. As a result, node z may need to wait for its neighbors, and so forth, to enter S_2, and thereby creating a deadlock. This, however, is impossible because there must be one or more nodes with the maximum Δ value. If these nodes are next to each other, then the one with a lower ID will transmit. Consequently, one of the nodes with the highest Δ value will move itself to S_2, causing its neighbors' Δ value to decrease. As a result, nodes will find either they have the highest Δ value, and thereby move into S_2 or their neighbors will move into S_2; in both cases, their Δ value decreases, and thereby proving the lemma. □

Corollary 1. *Each node is guaranteed to move into the* **ScheduleEnd** *state.*

Proof. By Lemma 1, each node's Δ value will reach either zero or a negative value at some time t. At such time t, the node will enter the **ScheduleEnd** state, which proves the statement. □

Lemma 2. *In each time slot, Algo-d will always activate links that have not been activated in previous slots.*

Proof. After generating the schedule for slot t, each node will remove links activated in slot t. If there are links yet to be activated, Algo-d will generate the schedule for slot $t+1$ using only these links. According to the state diagram, see Figure 3.3, Algo-d will not stop unless all links are scheduled. Thus Algo-d will always activate each link at least once. □

Proposition 2. *The schedule generated by Algo-d conforms to the* no-Tx-Rx *constraint.*

Proof. In each time slot, Algo-d places nodes in two different sets S_1 and S_2. Nodes in S_2 transmit and nodes in S_1 receive. Initially all nodes are in S_1 and Algo-d moves eligible nodes to S_2. Thus each node must belong to either S_1 or S_2. While a node moves itself from S_1 to S_2, it will send a message telling all its neighbors that it will be in S_2. Thus each node is aware which set its neighbor belongs to. As only those nodes that belong to S_2 transmit and nodes belonging to S_1 receives in each time slot, the resulting schedule thus is valid and follows the *no-Tx-Rx* constraint. □

The complexity of Algo-d, which is how many "rounds" it takes for Algo-d to generate a schedule, is analyzed next. Define a *round* as the time period starting from a node's first message exchange with its neighbors and ending with a neighbor moving to S_2. Define Δ_{in} as the maximum incoming degree. Similarly, define Δ_{out} as the maximum outgoing degree. If each node has a maximum of γ antennas/radios, then $\Delta_{in} = \Delta_{out} = \gamma$. Recall that E is the set of edges. The analysis of Algo-d's property in line and cycle topology is presented first before extending to general topologies.

Proposition 3. *For a line or cycle topology with n nodes, Algo-D needs at most $\frac{n}{2}$ rounds to schedule all links if n is even. Otherwise, it needs $\frac{n-1}{2}$ rounds.*

Proof. In a line or cycle topology, the maximum node degree is two. In the worst case, only one node moves into S_2 in each round. Thus a topology with n nodes needs at most $\frac{n}{2}$ rounds, if n is even, or $\frac{n-1}{2}$ rounds, if n is odd for Algo-d to schedule all links. □

Proposition 4. *For an arbitrary topology, in each time slot, all edges will be scheduled within* $R = \frac{|E|}{\Delta_{in}+\Delta_{out}-1}$ *rounds.*

Proof. Consider a directional edge $e_{a,b}$. If it is selected, then all incoming edges of node a and all outgoing links of node b are blocked from being activated due to the *no-Tx-Rx* constraint. Consequently, activating each edge blocks $(\Delta_{in} + \Delta_{out} - 1)$ edges or removes said number of edges from E. This means in the worst case after $\frac{|E|}{\Delta_{in}+\Delta_{out}-1}$ rounds, all edges will either be scheduled or blocked. □

Corollary 2. *The maximum number of transmitting links scheduled is bounded by* $R\Delta_{out}$

Proof. Whenever an edge $e_{a,b}$ is scheduled for transmission, all outgoing links of node a can be scheduled for transmission as well. This means there are up to Δ_{out} edges in each of the R groups of edges. Thus, the maximum number of transmitting links is $R\Delta_{out}$. □

The number of message exchanges for each node is presented next. Define one *message exchange* to be the successful transmission of a message from node i to all its neighbors.

Proposition 5. *For an arbitrary topology, the maximum number of message exchanges to generate the schedule for one slot by a node* i *is* $\frac{\Delta_i}{2} + 1$.

Proof. Node i will exchange a Txy message with all its neighbors at the start of each round. In each round, in the worst case, only one of node i's neighbors moves to S_2, meaning it will take $\frac{\Delta}{2}$ Updxy message exchanges for Δ_i to be reduced to zero or negative. We therefore have $\frac{\Delta_i}{2} + 1$ messages to generate the schedule for one slot. □

Proposition 6. *The number of time slots used to schedule all links is bounded by the node degree.*

Proof. Consider a network consisting of n nodes with a maximum degree $|N_v|$. In the worst case, only one neighbor of v moves to S_2 each slot, meaning it takes $|N_v|-1$ slots for node v to activate all its links. Thus, the number of time slots to activate all links is bounded by the node degree. □

The following proposition pertains to the "RootRoute" protocol outlined in Section 3.3.4.

Proposition 7. *RootRoute guarantees only the node with the lowest ID sends out the StartSFSlot message. In other words, only one node will issue the said message.*

Proof. First recall the fact that a node will only respond to its parent upon receiving a FinishSched message from *all* its children. Consider a non-root node N_z that transmitted a IRoot message to its children. As the network is connected, one of the children, say N_i, will have a route to a two hop neighbor of the root or the root itself; say N_j. As N_j knows of a node with a lower ID, it will discard the IRoot message. This implies that nodes $N_z, N_i \ldots, N_j$ cannot respond to their respective parent. Thus, N_z will not send out a StartSFSlot message because it is waiting for a FinishSched message from N_i. □

3.5 Evaluation

Algo-d's performance is evaluated in Matlab using the Matgraph [106] toolkit. In these experiments, assume all nodes are stationary, randomly connected and each node has a radio/beam for each neighbor. There are two sets of evaluation. First, the number of nodes varies from 10 to 70. Each node establishes a bidirectional link to another node with probability 0.5. Second, the number of nodes is fixed to 40 and the degree of each node varies from three to seven.

This thesis compares Algo-d with Algo-1 [30], Algo-2 [31], ROMA [24] and Jazzy-Mac [25]. Briefly, these algorithms work as follows:

- *Algo-1*, a centralized scheduler, recursively generates a MAX-CUT in every other slot. Nodes that transmit in slot t_i become receivers in slot $t_i + 1$.
- *Algo-2*, a centralized scheduler, generates a MAX-CUT in every slot. Nodes in one set transmit and nodes in the other set receive. It then removes activated links and generates a new MAX-CUT for the next time slot.
- *JazzyMAC*, a distributed scheduler, assigns each link a token. A node transmits only when it has the token for all its links.
- *ROMA*, a distributed scheduler, uses a hash function together with node ID, current time slot and link weight to schedule links without signaling messages. ROMA works in a pseudo-random manner.

The shown results are an average of 20 simulation runs. Each simulation run uses a different topology. In each experiment, the following metrics are computed:

- *Superframe length*, which corresponds to the number of slots required to activate all links at least once.
- *Network capacity*. This is the average number of links activated in each time slot, and is calculated by summing the number of links activated in each slot and dividing the resulting sum by the superframe length.
- *End-to-end delay*. This is the average end-to-end delay of randomly picked end-to-end routes. This is calculated by summing the delay of routes picked and dividing the resulting sum by the number of routes.

The resulting figures include the confidence interval, where 95% of the results are within the indicated error bars.

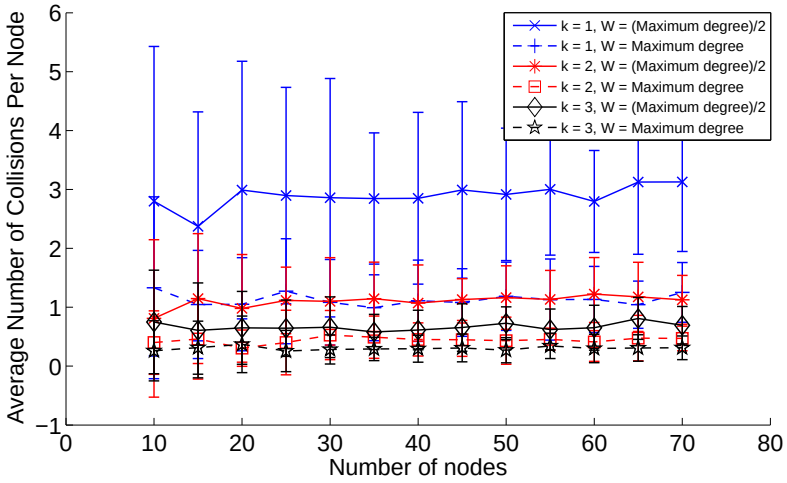


Figure 3.7: Average number of collisions experienced by a node under various node density.

3.5.1 Average Collision Per Node

Proposition 1 shows that increasing k and W reduces the probability of collisions. To validate this proposition, Figures 3.7 and 3.8 show the average number of collisions experienced by nodes for varying node densities and degrees. We see that, as expected, the number of collisions experienced by each node decreases with increasing k and W. When $k = 1$, each node experiences more than one collision. More importantly, the number of collisions experienced by a node does not increase with increasing node densities or degrees. From Figure 3.8, when kW is larger than the maximum node degree, each node experiences two to five collisions on average while deriving a schedule. Recall that Algo-d will not schedule a node's links in a given slot if the node experiences more than Φ collisions. This will result in a longer superframe and fewer number of links activated in each slot. Thus all experiments in this chapter only consider the situation when kW is larger than the maximum node degree. From these results, all following experiments use $\Phi = 3$ as the maximum number of retransmissions performed by each node to ensure a high probability of success and minimal channel access delays.

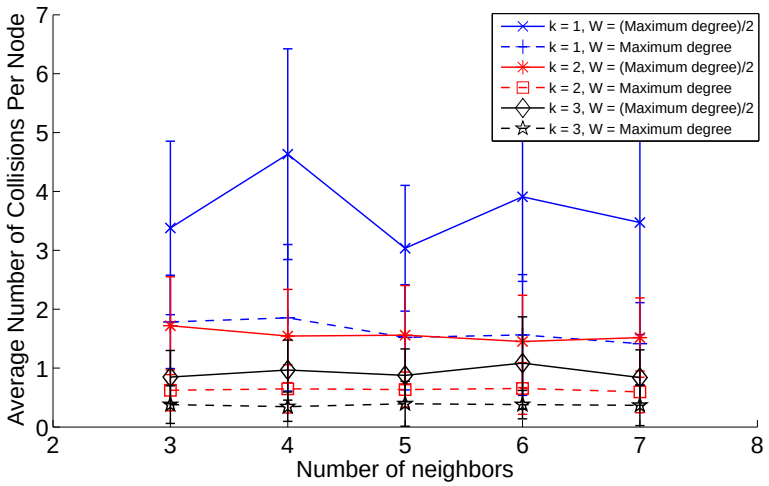


Figure 3.8: Average number of collisions experienced by a node under various node degrees.

3.5.2 Schedule Construction Cost

The experiments conducted in this section investigate the cost associated with schedule construction. Specifically, the total number of message exchanges and the number of slots required to generate a schedule for different k and W values. The network size is fixed to 40, and the node degree varies from 3 to 7. This thesis only studies varying degrees because delays are mainly affected by contention access, which is directly proportional to node degree.

Figure 3.9 shows the average number of message exchanged by each node. For different k and W, the results are similar. This is because when $\Phi = 3$, the transmission success probability is comparable to the different k and W values experimented. Figure 3.10 shows the number of slots used to derive a link schedule under different node degrees. As mentioned in Section 3.1, nodes are *static*. This means the schedule is generated infrequently; e.g., whenever there is a topological change such as the addition of a new router. Consequently, the cost of generating the schedule is amortized over many time periods. Having said that, the cost of generating the schedule remains low. Referring to Figure 3.10, when the network size is 40 and each node has seven neighbors, the time to generate the schedule is about 230 slots. Assuming IEEE 802.11ac radios with a physical link rate of 800 Mbps, and a packet

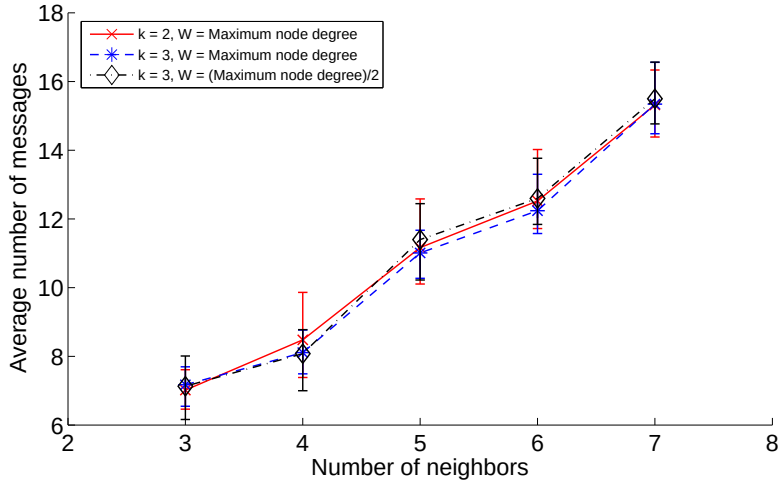


Figure 3.9: Average number of message exchanged per node to generate a schedule under various node degrees.

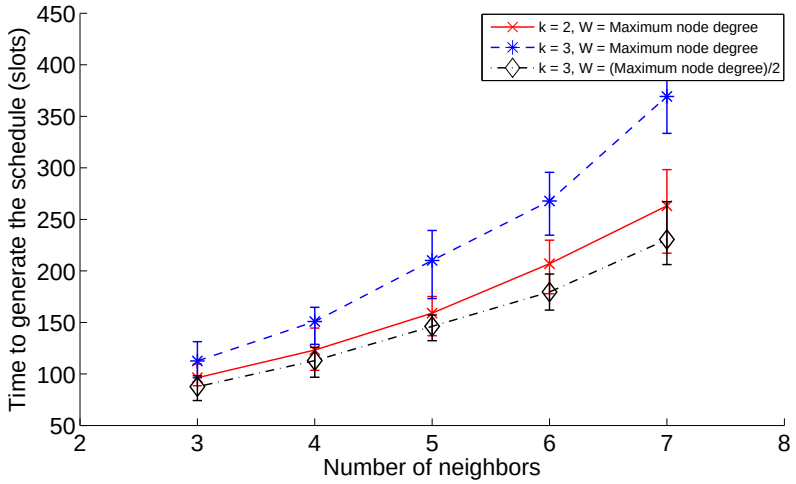


Figure 3.10: Time to generate a schedule under various node degrees.

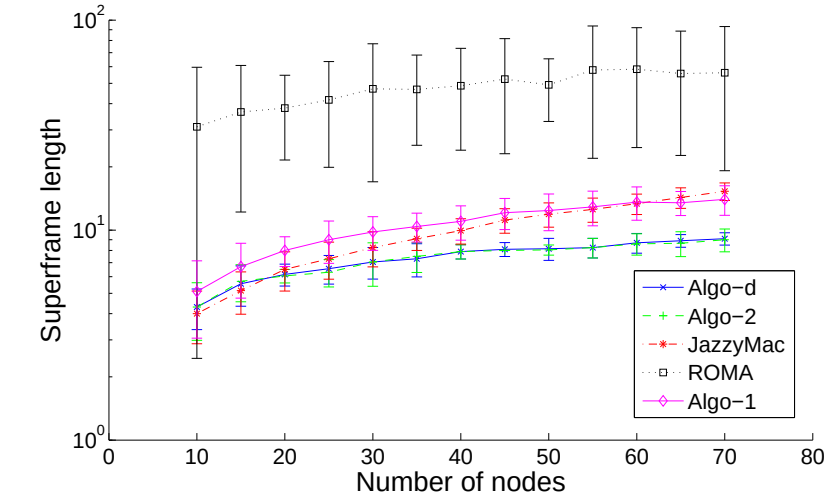
length of 800 bits, the slot size to transmit one packet is one microsecond. This means the total cost to generate the schedule is 230 slots, which is less than one millisecond. To this end, base on these results, for the remaining experiments, k is set to 3 and W is set to half the maximum node degree.

3.5.3 Node Density

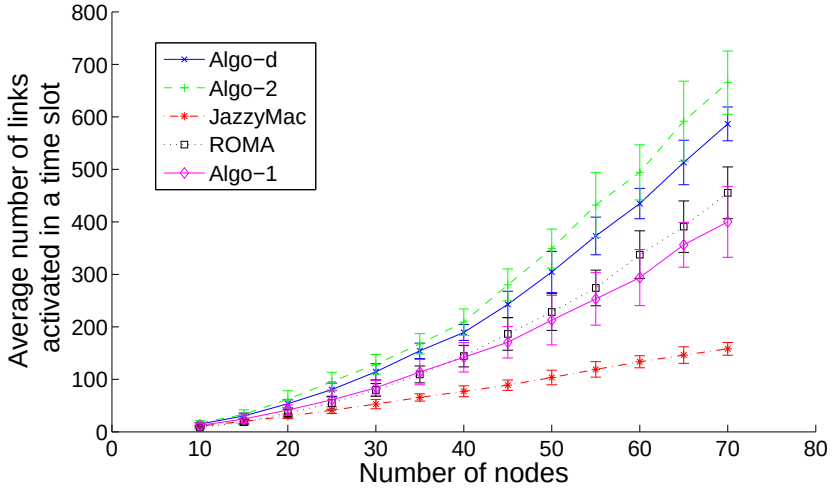
This section studies the impact of node density on superframe length, network capacity and average end-to-end delay. The algorithms are tested on topologies with 10 to 70 nodes. From Figure 3.11, we can see that Algo-1, Algo-2, JazzyMac and

Algo-d have similar superframe lengths when there are 10 nodes. The superframe length and the average number of links activated in a time slot increases with the number of nodes. However, for Algo-1 and JazzyMac, their superframe lengths increase much faster than Algo-2 and Algo-d. When the network size is 70, the superframe length of Algo-1 and JazzyMac is about 55% longer than the superframe length generated by Algo-d. For ROMA, the superframe length is about 6-7 times that of Algo-d. From Figure 3.11b, we can see that Algo-2 and Algo-d schedule more links in each slot than Algo-1, ROMA and JazzyMac. When the network size is 70, Algo-d schedules 12% fewer links than Algo-2, 28% more links than ROMA, 46.5% more links than Algo-1 and 270% more links than JazzyMac. The performance of Algo-d shown in Figure 3.11b includes opportunistic links as mentioned in Section 3.3. With this optimization, Algo-d schedules 46.7% to 119% more links when the network size is 10 and 70 respectively. The increment becomes larger as the network size increases. Figure 3.11c indicates that Algo-d, Algo-1 and Algo-2 have similar end-to-end delays. At the same time, ROMA's end-to-end delay is 3-4 times that of Algo-d and for JazzyMac, the end-to-end delay is 50% less than Algo-d.

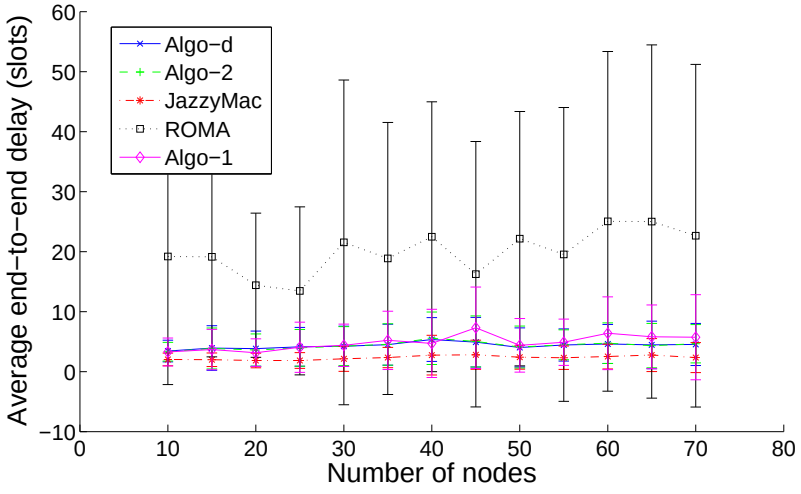
The reasons for the aforementioned performance are due to the following reasons. Algo-2 moves a node to the other set if moving this node generates more links between the two sets. In addition, it balances the number of nodes between the sets if doing so does not decrease the number of links. However, Algo-d cannot use these rules to balance the number of nodes in each set as Algo-d is distributed and each node cannot count the total number of nodes in each set without incurring excessive signaling overheads. Also, Algo-2 adds opportunistic links in the following manner. After deriving the schedule for the current time slot, Algo-2 creates a new topology by removing all scheduled links. Then Algo-2 generates another schedule for the newly created topology to add opportunistic links. This deterministic process of Algo-2 adds more opportunistic links than the random process used by Algo-d. Unfortunately, Algo-d cannot apply the same process as Algo-2 because generating another schedule incurs expensive signaling overheads. Further, in the scenario



(a)



(b)

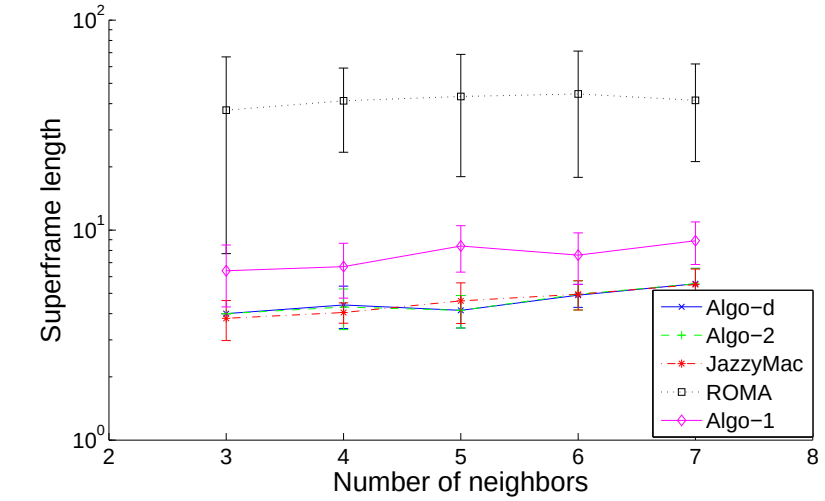


(c)

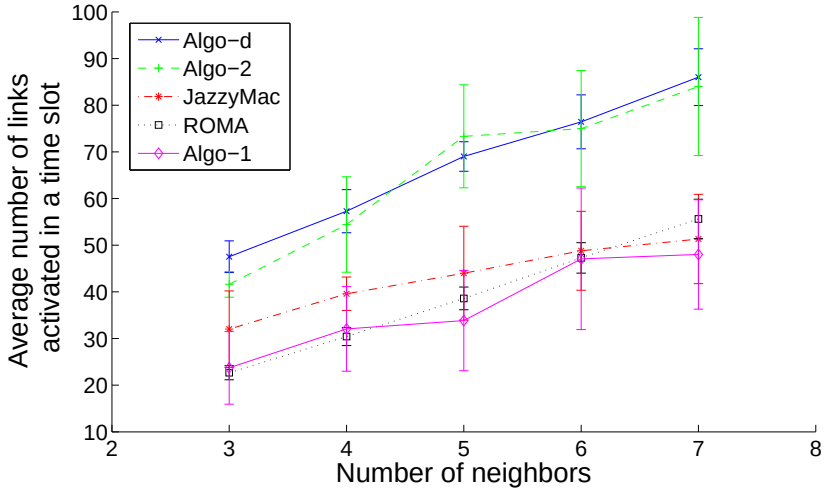
Figure 3.11: Performance under different node densities

whereby Algo-2 does not balance nodes in each set and both Algo-d and Algo-2 do not add opportunistic links. Interestingly, both algorithms generate the same link schedule.

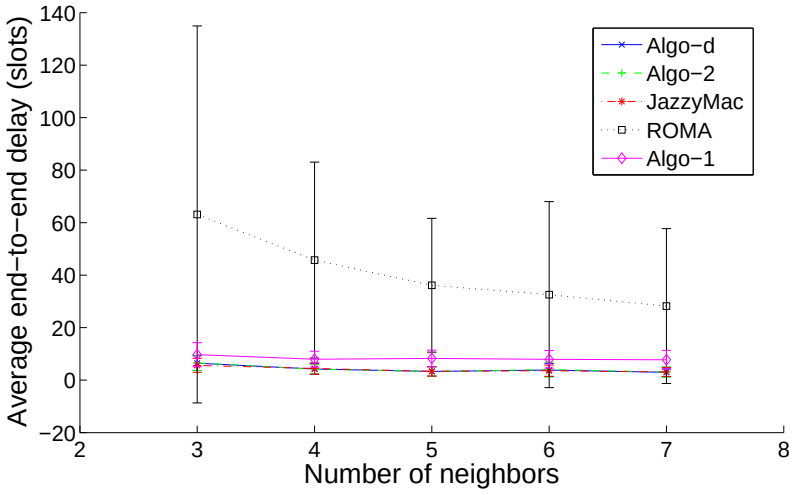
Recall that Algo-1 uses a 2-phase transmit/receive scheme to generate a link schedule every other time slot. Nodes transmitting in one time slot will be receiving in the next time slot; the second time slot is a mirror of the previous. This obviates any opportunities to add links that do not conflict with the transmissions in the second slot. Consequently, the 2-phase based algorithm results in an inefficient slot usage and longer superframe length. As shown in [31], generating a MAX-CUT in a slot-by-slot manner yields more links and shorter superframe lengths. The poor performance of JazzyMac is due to the fact that a node must wait until it holds the token of all its links before it can start transmission. This results in a large number of idle links as their end nodes are waiting for tokens. Thus the superframe length increases linearly and the number of links activated grows 2.6 times slower than Algo-1 and four times slower than Algo-d. In Figure 3.11c, we see that nodes using JazzyMac incur 50% less delay than Algo-d. This is because for JazzyMac, if there are many packets queued at a node, it will transmit all packets before releasing the corresponding tokens. However for Algo-d, every link gets one slot within one superframe, meaning the node will take several superframes to empty the same number of packets. Note that this behavior of JazzyMAC is unfair and in fact reduces network capacity as a heavily loaded link may block others from transmitting. In ROMA, each node determines whether it is in transmit or receive mode in a random manner. However, nodes that have links yet to be activated do not have a higher priority to become a transmitter unless these links have a long queue. As a result, ROMA has the longest superframe length and end-to-end delay.



(a)



(b)



(c)

Figure 3.12: Performance under different node degrees

3.5.4 Node Degree

This experiment is similar to the previous one except that the number of neighbors is the variable. The network density is fixed at 40 nodes while the node degree varies from 3 to 7. Results are shown in Figure 3.12. From Figure 3.12 we can see that Algo-d, Algo-2 and JazzyMac have similar superframe length in all experiments and Algo-1 does not perform well. Algo-1's superframe length is about 1.35 times of Algo-d. Due to ROMA's random scheme, it has a long superframe, which is 3.5 times that of Algo-d. Algo-d schedules 2% more links than Algo-2, 68% more links than JazzyMAC, 55% more links than ROMA and 79% more links than Algo-1 when each node has seven neighbors. Algo-1's two-phase transmit/receive scheme leads to a longer superframe length and a wider confidence interval. As mentioned earlier, nodes in JazzyMac are only able to transmit when they hold all tokens. Consequently, there are fewer links activated per slot.

3.6 Conclusion

This chapter has studied the problem of deriving a superframe with minimal number of slots and maximal number of links in each slot. To address the said problem, this chapter presents Algo-d, the first distributed solution to the MAX-CUT problem. Through comprehensive evaluation, it is shown that Algo-d has similar performance to Algo-2, the state-of-the-art centralized algorithm in terms of superframe length and the number of concurrent links in each slot. Moreover, Algo-d schedules 28%, 46.5% and 270% more links in each slot as compared to ROMA, Algo-1 and JazzyMac respectively.

A key observation is that Algo-d only considers single hop packet transmissions; i.e., a node has one packet to be transmitted to a neighbor. However, in infrastructure networks, data forwarding to/from gateway nodes over multiple hops is a fundamental operation. In addition, a key aim is to ensure these packets are forwarded to/from one or more gateway nodes in minimum time. This is the topic of

the next chapter.

Chapter 4

On Personalized Broadcast and Forest Construction in Multi Tx/Rx Wireless Mesh Networks

To date, existing personalized broadcast schedulers or data collection approaches do not consider the MTR capability of nodes. They do not consider the possibility of sending multiple packets to the same neighbor using multiple antennas. Thus, they are unlikely to yield a minimal personalized broadcast length for use in MTR WMNs. Henceforth, this chapter presents a novel personalized broadcast scheduler that aims to generate a short personalized broadcast schedule in MTR WMNs. In particular, it aims to design a scheduler that utilizes the spatial multiplexing capability of nodes. Moreover, it considers suppressing neighboring interference (NI). Apart from that, this chapter considers using multiple gateways to further reduce the personalized broadcast makespan. It addresses the forest construction problem using an ILP and a heuristic solution. Experimental results show that the proposed scheduler, Algo-PB, outperforms the state-of-the-art link scheduler proposed in [26] in terms

schedule length. In addition, compared with the ILP, Algo-FC has lower complexity and generates a near-optimal balanced forest.

This chapter is organized as follows. Section 4.1 presents the network model and describes the personalized broadcast problem and the forest construction problem. Section 4.2 analyzes the theoretical lower bound of the personalized broadcast makespan. Solutions for the personalized broadcast problem and forest construction problem are outlined in Section 4.3 and 4.4, respectively. The characteristics of the proposed algorithms are analyzed in Section 4.5. Experimental results are presented and discussed in Section 4.6. The conclusion is presented in Section 4.7.

4.1 Preliminaries

Consider an MTR-WMN represented as a graph $G(V, E)$ where V denotes the set of static vertices/nodes/routers and E denotes the set of directional links; i.e., link $e_{v,u} \in E$ denotes a link from node $v \in V$ to $u \in V$. Each node $v \in V$ has a transmission range R_t. Links $e_{v,u}$ and $e_{u,v}$ exist if the distance between node v and u is within R_t. Assume a node's interference range R_i is equal to its transmission range; i.e., $R_t = R_i$. In addition, assume nodes have the channel state information (CSI) of their neighbors. This is reasonable as mesh routers are fixed and they can send out pilot symbols periodically to obtain the required CSI [70]. Each node $v \in V$ has Δ radios/antennas or Degree of Freedom (DoF). The number of antennas used by node v for transmissions, receptions and NI cancellation is denoted as Δ_v^+, Δ_v^- and Δ_v^*, respectively. Note that the end nodes of a link must dedicate an antenna whenever it transmits or receives a packet. Let $\mathcal{I}_v$ be the set of neighbors of a transmitter v that are receiving at least one packet from their neighbor(s) except node v. Then node v must cancel any NI it causes to nodes in $\mathcal{I}_v$. According to [28], for each node $u \in \mathcal{I}_v$, node v needs to use Δ_u^- antennas to cancel NI. We thus have the following NI cancellation (NIC) constraint:

Constraint 1. A transmitting node v must cancel all NI it causes to its neighboring

receivers, meaning it must dedicate $\Delta_v^* = \sum_{u \in \mathcal{I}_v} \Delta_u^-$ antennas for NIC.

The following constraint bounds the number of antennas used by a node:

Constraint 2. The total number of antenna elements that a node uses for data transmission/reception and NIC must not exceed the total number of antenna elements it has, i.e., $\Delta_v^+ + \Delta_v^- + \Delta_v^* \leq \Delta$.

As the third constraint, each node must follow the *no-tx-rx* or *half-duplex* requirement. Formally,

Constraint 3. $\Delta_v^+ \times \Delta_v^- = 0$.

Let $S \subset V$ be a set of nodes designated as gateways. For a given node $v \in V - S$, let $r_v \geq 0$ be the number of requests/packets to be received from one of the gateways. Assume that in the resulting schedule, each non-gateway node receives packets from only one gateway. Let T^s be the tree rooted at gateway $s \in S$ and $V_s \subset V$ be the set of descendants of gateway s. Note that V_s contains only non-gateway nodes. Formally, $w_v^s = a \times r_v + b \times dist(v, s)$ denotes the weight of node $v \in V_s$; see Section 4.4 for more details. Here $dist(v, s)$ is the distance (in hops) between node v and its gateway s, while a and b are coefficients that are set depending on the value of Δ. Let $\mathcal{W}_s = \sum_{v \in V_s} w_v^s$ be the weight/load of gateway s. Assume time is slotted where each slot corresponds to the transmission of one packet. Here, the required synchronization can be achieved using a GPS module.

The formal definition of the personalized broadcast problem and the forest construction problem are as follows.

Definition 3. For a given T^s, the *personalized broadcast problem* asks for a collision-free link schedule with the minimal makespan, called *personalized broadcast schedule*, that allows gateway s to transfer $r_v \geq 0$ packets to each destination node $v \in V_s$ subject to constraints 1, 2 and 3.

Definition 4. For a given MTR-WMN $G(V, E)$ with $|S|$ gateways, the *forest construction problem* aims to build a forest containing $|S|$ trees, i.e., a set of T^s for each

gateway $s \in S$, such that the maximum makespan among all personalized broadcast schedules in G is minimized.

Remark that the personalized broadcast problem is NP-complete. The authors in [26] proved the NP-hardness of the personalized broadcast problem by reduction from the well-known Partition Problem. Specifically, the decision version addressed in [26] is as follows: given a network $G(V, E)$, integer weights $r_v \geq 0$, where $v \in V_s$, and an integer bound K, is there a routing tree in G and a multi-hop personalized broadcast schedule that transmits r_v packets from a gateway s to each node v in a collision free manner and has a makespan less than K? In their network model, only one transmission is allowed at a node at a time; i.e., $\Delta = 1$. This thesis generalized this NP-complete problem by considering $\Delta \geq 1$.

4.2 Upper and Lower Bound

This section analyzes the theoretical bounds of the personalized broadcast schedule for a tree rooted at s; i.e., T^s, for $s \in S$. The upper bound of the schedule is analyzed first. Denote L to be the number of levels or height of tree T^s, and D_l to be the total packets destined for level l of T^s, where $1 \leq l \leq L$. Assume gateways are at level zero. This section then makes the following propositions.

Proposition 8. *The schedule length for a tree T^s is upper bounded by (i) D_1 slots, for $L = 1$, (ii) $D_1 + 2D_2$ slots, for $L = 2$, or (iii) $D_1 + 2D_2 + \sum_{l=3}^{L} l + 3(D_l - 1)$ slots, for $L \geq 3$.*

Proof. The worst case is when all of the following conditions apply: (a) gateway s transmits packets on a level-by-level basis, i.e., gateway s first transmits all packets for nodes at level L, followed by those for nodes at level $L - 1$, and so forth, (b) each node can transmit only one packet at a time, and (c) any transmitting node at level $l \geq 1$ interferes with all nodes at level $l - 1$, l and $l + 1$.

Start with $L = 1$. Following (b), as the gateway s transmits one packet in each

slot, it will require D_1 slots; the proposition is thus true for $L = 1$. Next, consider $L = 2$. For this case, as per (c), a transmitting node at level $l = 1$ interferes with all nodes at level 1 as well as nodes at level 2. Thus, the first packet for nodes at level 2 arrives in slot 2, the second packet arrives in slot $2 + 2 = 4$, the third packet arrives at level 2 in slot $2 + 2 + 2 = 6$, and so forth. The last packet arrives in slot $2D_2$. This means packets for nodes at level $l = 2$ require $2D_2$ slots. As the gateway uses D_1 slots to deliver packets destined for nodes at level $l = 1$, part (ii) of the proposition is thus true.

For case (iii) of the proposition, first consider $L = 3$. The first packet from gateway s arrives at level $l = 3$ in slot 3. In this case, as per (c), a transmitting node at level $l = 2$ interferes with all nodes at level 1, 2 and 3. Thus, the second packet to level $l = 3$ arrives in slot $l + 3 = 6$, the third packet arriving in slot $l + 6 = 9$ and so forth. Consequently, transferring D_3 packets to level $l = 3$ requires no more than $3D_3$ slots. In general, when a node at level l is receiving a packet from a node at level $l1$, the next packet destined to any node at level l can be arriving at a node at level $l3$ without causing interference, and thus the packet needs 3 additional slots to arrive at level l. Consequently, for any level $l \geq 3$, after the first packet that requires l slots, each subsequence packet arrives at level l every 3 slots. Thus, packet transmission to a node at level $l \geq 3$ is upper bounded by $l + 3(D_l1)$ slots. Lastly, as D_2 and D_1 require $2D_2$ and D_1 slots, the schedule upper bound is at most $D_1 + 2D_2 + \sum_{l=3}^{L} l + 3(D_l - 1)$ slots. This proves case (iii) of the proposition.

□

The next results concern the lower bound of the personalized broadcast schedule. Denote T_i^s to be a sub-tree of T^s rooted at the i-th child of s. Further, let $\delta(v, s)$ be the makespan of the schedule used to transmit r_v packets from s to v. We thus have the following propositions. Note that the propositions assume there is no NI among nodes and thus constitute the best case scenario. The lower bound in the presence of NI remains an open question.

Proposition 9. *The makespan lower bound to transmit $r_v > 0$ packets from s to v for T^s is (i) $\delta(v,s) \geq dist(v,s) + 2 \times \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots for $dist(v,s) \geq 2$, or (ii) $\delta(v,s) \geq dist(v,s) + \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots, for $dist(v,s) = 1$.*

Proof. First consider the case where $dist(v,s) \geq 2$. Observe that the first Δ packets from s reach v no faster than $dist(v,s)$ slots. For $r_v \leq \Delta$, we have $\lceil \frac{(r_v - \Delta)}{\Delta} \rceil = 0$, and thus the proposition is true for this case. However, for $r_v > \Delta$, the *no-tx-rx* constraint must be taken into consideration. Specifically, any node in the path from s to v can only transmit or receive up to Δ packets at a time. Thus, v receives the next Δ packets no earlier than two slots after it receives the first Δ packets. In general, in the best case, v can receive up to Δ of the remaining $r_v - \Delta$ packets every two slots. Thus, v receives the remaining $r_v - \Delta$ packets in no more than $2 \times \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots, giving a makespan of $dist(v,s) + 2 \times \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots. For case (ii), since v is only one hop away from s, it receives up to Δ packets every slot. Specifically, for $r_v \leq \Delta$, node v receives its r_v packets in $dist(v,s) = 1$ slot, while for $r_v > \Delta$, v receives all r_v packets in $dist(v,s) + \lceil \frac{(r_v - \Delta)}{\Delta} \rceil$ slots, proving the proposition for case (ii). □

Let L_i be the number of levels in sub-tree T_i^s, and D_i^l be the total number of packets to be transmitted to level l of sub-tree T_i^s. Define $\mathcal{F}_i^l$ to be the *last slot* in which a node at level l of sub-tree T_i^s receives its last packet. For example, in network shown in Figure 4.1, suppose node G and H on level 3 of subtree T_A^s receive their last packet at slot 4 and 5 respectively, thus $\mathcal{F}_A^3 = 5$.

Proposition 10. *For any node v at level l in a T_i^s and $D_i^l > 0$, (i) $\mathcal{F}_i^l = l + 2 \times \lceil \frac{D_i^l - \Delta}{\Delta} \rceil + 2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$, for $l \geq 2$, or (ii) $\mathcal{F}_i^l = l + \lceil \frac{D_i^l - \Delta}{\Delta} \rceil + 2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$, for $l = 1$.*

Proof. For a node v at level l and the i-th child of gateway s, applying Proposition 9, one can obtain (i) $\delta(v,i) \geq l + 2 \times \lceil \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rceil$ slots for $l \geq 2$, or (ii) $\delta(v,i) \geq l + \lceil \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rceil$ slots, for $l = 1$. Node v will first relay the packets destined to its descendants. This incures $2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$ slots. After that, it starts receiving its

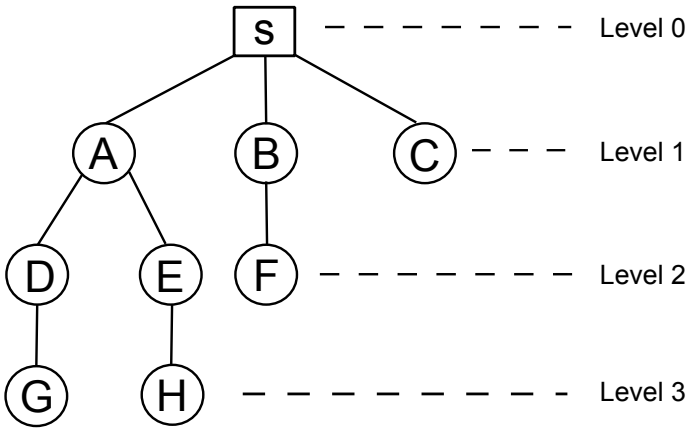


Figure 4.1: Example topology

first packet from the gateway. Each of the relayed packets takes two slots because the gateway transmits at an interval of two slots due to the *no-tx-rx* constraint. The last busy slot, i.e., $\mathcal{F}_i^l$, for level l of sub-tree rooted at i is thus the summation of the time required to relay descendants' packets plus the time to receive all its packets. □

As a result of Proposition 9 and 10, if $r_v = 0$, the number of slots for node v to receive packets located at itself is 0. Also, if $D_i^l = 0$, the number of slots needed to transfer packets to level l of sub-tree T_i^s is zero.

Proposition 11. *The makespan of the personalized broadcast schedule for a sub-tree T_i^s is lower bounded by $\mathcal{F}_i = MAX(\mathcal{F}_i^l)$ for $l \in 1, \ldots, L_i$.*

Proof. According to Proposition 10, any node v at level l of sub-tree T_i^s has $\mathcal{F}_i^l$. This implies that the last busy slot of sub-tree T_i^s cannot be earlier than $MAX(\mathcal{F}_i^l)$, for $l \in 1, \ldots, L_i$. □

Proposition 12. *The makespan of the personalized broadcast schedule for a tree T^s with θ sub-trees is lower bounded by $MAX(\mathcal{F}_{max}, \lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil)$, where $\mathcal{F}_{max} = MAX(\mathcal{F}_i)$ for $i \in [1, 2, \ldots \theta]$.*

Proof. Without loss of generality, the first sub-tree of T^s, i.e., T_1^s, produces $\mathcal{F}_{max}$. According to Proposition 10, the root of each sub-tree T_i^s will first receive packets located at its sub-tree and forward them onwards. This means the gateway node

s cannot serve the same sub-tree for two continuous slots due to the *no-tx-rx* constraint. Consequently, one can *interleave* the transmissions of two sub-trees. For example, node s transmits to sub-tree T_1^s in slots 1, 3, 5, and so forth and transmits to sub-tree T_2^s in slot 2, 4, 6, ... and so forth. Proposition 11 shows that $\mathcal{F}_i$ is the total number of slots needed to deliver all packets to sub-tree T_i^s. This implies the following facts:

- Case 1: $\mathcal{F}_1 > \mathcal{F}_2 + \mathcal{F}_3 + \ldots + \mathcal{F}_\theta$. All other sub-trees can be interleaved with the transmissions to T_1^s. That is, the transmissions to all sub-trees T_2^s to T_θ^s will finish earlier than $\mathcal{F}_1$, thus the total number of slots needed to deliver all packets to nodes in T is lower bounded by $\mathcal{F}_1$, i.e., $\mathcal{F}_{max}$.

- Case 2: $\mathcal{F}_1 \leq \mathcal{F}_2 + \mathcal{F}_3 + \ldots + \mathcal{F}_\theta$. In this case, all other sub-trees cannot be fully interleaved with the transmission to T_1^s. Thus, this thesis proves a *loose* lower bound here, i.e., only considers the number of slots needed for the gateway to transmit all packets to the nodes located at the first level, which is $\lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil$. Thus, the number of slots needed to deliver all packets will not be shorter than $\lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil$.

Note that the schedule makespan will not be shorter than either $\mathcal{F}_{max}$ or the number of slots needed for the gateway to inject all packets into the network, and thus the personalized broadcast is lower bounded by $MAX(\mathcal{F}_{max}, \lceil \frac{\sum_{v \in V_s} r_v}{\Delta} \rceil)$. □

A brief example and how the theoretical lower bound is calculated are shown as follows. For the topology shown in Figure 4.1, assume s is the gateway and each node has one packet; $r_v = 1$ for $v \in V_s = \{A, B, C, D, E, F, G, H\}$. Denote the sub-tree rooted at node A, B and C as T_1^s, T_2^s and T_3^s, respectively. Assume each node has $\Delta = 2$ antennas.

First focus on sub-tree T_1^s. The total number of packets destined at level 3 of sub-tree T_1^s is $D_1^3 = 2$, and nodes located at level 3 do not need to forward packets as there are no nodes located beyond level 3. According to Proposition 10, it is

easy to obtain that $\mathcal{F}_1^3 = 3$. Now consider level 2 of T_1^s. Nodes at level 2 need to forward packets to nodes located at level 3 first. For nodes located at level 2, they need to forward packets located at level 3 before receiving packets located at level 2. As $D_1^3 = 2$, nodes located at level 2 need to forward packets to level 3 $\lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$ times which incurs $2 \times \lfloor \frac{\sum_{m=l+1}^{L_i} D_i^m}{\Delta} \rfloor$ slots. Then it is able to calculate that $\mathcal{F}_1^2 = 4$. Similarly, $\mathcal{F}_1^1 = 5$. According to Proposition 11, the lower bound for sub-tree T_1^s is $\mathcal{F}_1 = 5$. Applying the same procedure for sub-tree T_2^s and T_3^s results in $\mathcal{F}_2 = 2$ and $\mathcal{F}_3 = 1$. According to Proposition 12, as $\mathcal{F}_{max} = \mathcal{F}_1 > \mathcal{F}_2 + \mathcal{F}_3$, the lower bound of tree T^s is $\mathcal{F}_{max} = 5$.

4.3 Personalized Broadcast Scheduling

This section presents a centralized link scheduling algorithm to solve the aforementioned personalized broadcast problem. Note, a centralized solution is ideal for the current setting because information such as the topology and buffered packets are located conveniently at the gateway. The proposed algorithm, called Algo-PB, generates the schedule in a path-by-path manner. Its key idea is to always schedule one packet to the farthest node that has not received all its packets. It then labels each link along the path from the gateway to the farthest node using a modified version of Conflict Free Coloring (CF-Coloring) proposed in [107]. Briefly, given a path p, the modified coloring method designed in this thesis, Collision Free Link Upgrade Coloring (CFLU-Coloring), assigns an increasing positive integer to each link along p starting from the source node. Moreover, CFLU-Coloring considers both MTR and spatial multiplexing. When CFLU-Coloring colors node v's adjacent links, each color, represented by a natural number, can be used at most Δ times. The coloring result must follow constraints 1, 2 and 3. Further, the number/color assigned to each link along path p is strictly increasing to ensure the final schedule is conflict-free. Specifically, for each path p, CFLU-Coloring generates a tuple $< (e_{1,2}, c_1), (e_{2,3}, c_2) \ldots (e_{i,j}, c_i) >$, where $e_{i,j}$ denotes a link from node i to j

and c_i is the color/number assigned to link $e_{i,j}$. Links assigned color c_i are scheduled in the c_i-th slot, thus the maximum number of colors used in CFLU-Coloring is the makespan of the personalized broadcast schedule.

In the sequel, CFLU-Coloring is described using Algorithm 1 followed by a description of Algo-PB. Its input is a path p, and it outputs the tuple $Sched(p)$. CFLU-Coloring keeps an integer c that denotes the color CFLU-Coloring assigns to a link; see Line 1. Then CFLU-Coloring colors each link along path p, starting with the first link in p; see Line 2-8. Function $DoFCheck()$ is used to test whether assigning color c to link $e_{p(i),p(i+1)}$ satisfies constraints 1 to 3. If color c can be assigned to link $e_{p(i),p(i+1)}$, $DoFCheck()$ returns zero, otherwise $DoFCheck()$ returns one; see Line 3-5. After assigning color c to link $e_{p(i),p(i+1)}$, CFLU-Coloring updates $Sched(p)$; see Line 6. Line 7 of Algorithm 1 is used to ensure two consecutive links are not assigned the same color.

Algorithm 1: CFLU-Color()

```
   Input  : p
   Output: Schedule Sched(p)
1  c = 1, Sched(p) = ∅;
2  for i ← 1 to |p| − 1 do
3  |   while DoFCheck(p, i, c) do
4  |   |   c = c + 1 ;
5  |   end
6  |   Sched(p) = Sched(p) ∪ (e_{p(i),p(i+1)}, c);
7  |   c = c + 1;
8  end
```

Next, Algo-PB is presented in detail with the aid of Algorithm 2. To do this, this section makes a few notations and definitions. Let $\mathcal{V}$ be a sequence of nodes in decreasing distance order from gateway s. Let $Sched(p)$ be a tuple containing the output of CFLU-Coloring given the input p. Denote P as a collection of paths, and F is a set that records all the results from CFLU-Coloring; i.e., $F = \{Sched(p) \mid p \in P\}$. Let t_{max} be the maximum slot used in CFLU-Coloring; i.e., the makespan of the personalized broadcast schedule.

Algo-PB first sorts all nodes in V and stores them in $\mathcal{V}$ in decreasing distance

order from gateway s; see Line 1. Then Algo-PB sets a counter k to label each path and its schedule; see Line 2. Algo-PB works in rounds. In each round, Algo-PB picks the first node in $\mathcal{V}$, say v, and checks whether $r_v = 0$; see Line $4-5$. To aid the following discussion, v is used to denote the first node in $\mathcal{V}$ in each round. If $r_v = 0$, Algo-PB removes v from $\mathcal{V}$ and moves to the next round; see Line $5-7$. If $r_v \neq 0$, Algo-PB schedules one packet from the gateway to node v along the path p_k using CFLU-Coloring; see Line $9-10$. Then Algo-PB reduces r_v by one as it has scheduled one packet to v; see Line 11. The scheduling result is a tuple $Sched(p_k)$ and Algo-PB adds the result to F. After increasing k by one, Algo-PB moves to the next round; see Line $12-13$. When $\mathcal{V} = \emptyset$, meaning all packets are scheduled, the set F contains all links and their scheduled slots. Lastly, Algo-PB computes t_{max} and returns the resulting t_{max} and F; see Line 16 and 17 respectively.

Algorithm 2: Algo-PB

Input : V_s, r_u *for each* $u \in V_s$
Output: Schedule F, Schedule Length t_{max}
// Sort nodes in V_s in decreasing hop length order from s, and store them in tuple $\mathcal{V}$
1 $\mathcal{V} = Sort(V_s), F = \emptyset$;
// Label each path and its schedule
2 $k = 1$;
3 **while** $\mathcal{V} \neq \emptyset$ **do**
4 $\quad v = \mathcal{V}(1)$;
5 $\quad$ **if** $r_v = 0$ **then**
6 $\quad\quad \mathcal{V} = \mathcal{V} - v$;
7 $\quad\quad$ continue;
8 $\quad$ **else**
9 $\quad\quad p_k = path(s \rightarrow v)$;
10 $\quad\quad Sched(p_k) = \text{CFLUColor}(p_k)$;
11 $\quad\quad r_v = r_v - 1$;
12 $\quad\quad F = F \cup Sched(p_k)$;
13 $\quad\quad k = k + 1$;
14 $\quad$ **end**
15 **end**
// Function $makespan(F)$ will return the maximum color number c_i among all $(e_{i,j}, c_i) \in F$
16 $t_{max} = makespan(F)$;
17 return$(makespan(F), F)$;

Lastly, Figure 4.2 is now used to illustrate how Algo-PB works on a tree, where s is the gateway and each node has $\Delta = 2$ antennas. Denote a node set $V = \{A, B, C\}$. Suppose $r_A = r_B = 1$ and $r_C = 2$. Solid lines denote links in the network and dotted lines indicate interferences. Algo-PB first calculates the distance from the gateway s to each node and sorts nodes in V in decreasing hop length order. The sorted set can then be represented as $\mathcal{V} =< C, A, B >$; see Line 1. Figure 4.2 includes all four rounds of Algo-PB. Each arrow denotes the path picked in that round. Initially, the counter k is set to 1; see Line 2. In the first round, C is the first node in $\mathcal{V}$; thus $v = C$. As $r_C = 2$, Algo-PB then schedules a packet to C; see Line $8 - 14$. The path from the gateway s to node C is $p_1 = s \rightarrow A \rightarrow C$; see Line 9. Algo-PB then schedules p_1 using CFLU-Coloring. As p_1 is the first path to be scheduled, i.e., no color is used yet, the smallest positive available number is 1. Algo-PB colors link $e_{s,A}$ as 1 and link $e_{A,C}$ as 2; see Line 10. The result of CFLU-Coloring is $Sched(p_1) =< (e_{s,A}, 1), (e_{A,C}, 2) >$. Given that Algo-PB has scheduled one packet to C, it reduces r_C by one; see Line 11. It then adds $Sched(p_1)$ into F and increases k by one; see Line $12 - 13$. In the second round, C is still the first node in $\mathcal{V}$ and $r_C = 1$; thus Algo-PB schedules a packet to C along path $p_2 = s \rightarrow A \rightarrow C$. Assigning color 1 to link $e_{s,A}$ and color 2 to link $e_{A,C}$ satisfies constraints 1 to 3. Thus, the result of CFLU-Coloring on p_2 is $Sched(p_2) =< (e_{s,A}, 1), (e_{A,C}, 2) >$. Path $p_3 = s \rightarrow A$ is to be scheduled in the third round. Assigning color 1 to link $e_{s,A}$ does not satisfy constraint 2. This is because two of s's links are assigned color 1. Further, color 2 cannot be assigned to link $e_{s,A}$ because node A cannot transmit and receive simultaneously according to constraint 2. Thus, the result of CFLU-Coloring is $Sched(p_3) =< (e_{s,A}, 3) >$. Lastly, Algo-PB schedules a packet to node B along $p_4 = s \rightarrow B$. Color 1 cannot be assigned to link $e_{s,B}$ because it does not satisfy constraint 2. Further, Color 2 cannot be assigned to link $e_{s,B}$ either. This is because node B is interfered by transmitting node A and node A does not have enough antennas to cancel the interference it caused to node B. The result of CFLU-Coloring on p_4 is $Sched(p_4) =< (e_{s,B}, 3) >$. After scheduling all packets, Algo-PB

sets $t_{max} = makespan(F) = 3$ as the schedule length, see Line 16, and returns it together with $F =< (e_{s,A}, 1), (e_{A,C}, 2), (e_{s,A}, 1), (e_{A,C}, 2), (e_{s,A}, 3), (e_{s,B}, 3) >$ in Line 17 . Table 4.1 shows the resulting link schedule for the topology in Figure 4.2.

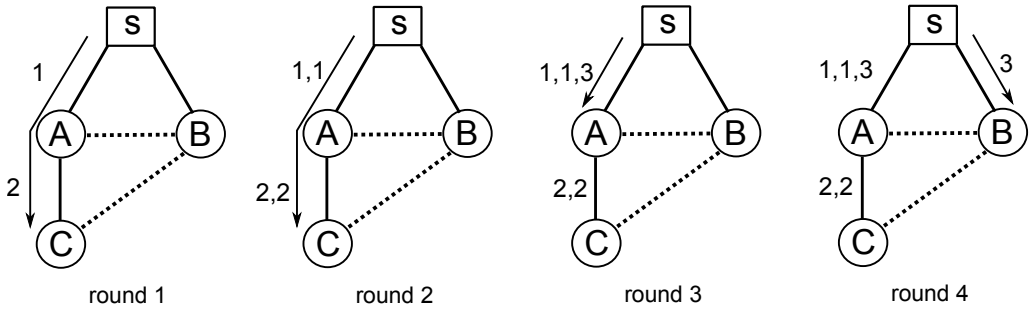


Figure 4.2: Example topology

Slot 1	*Slot 2*	*Slot 3*
$e_{s,A}$	$e_{A,C}$	$e_{s,A}$
$e_{s,A}$	$e_{A,C}$	$e_{s,B}$

Table 4.1: Link schedule for Figure 4.2

The gateway is responsible for informing nodes the latest schedule. This can be achieved by piggybacking the schedule in downstream packets. Also included is the start time of the schedule in which these slots take effect. Note, if there are no downlink packets, then a dummy packet can be created before the schedule is computed. The schedule generated by Algo-PB will thus include any dummy packets. Note that Algo-PB will be used in conjunction with a protocol that schedules packets in batches. This means any new packets that arrive when a schedule is in effect will not be transmitted in the current schedule; i.e., they will be scheduled in the next batch using a newly derived schedule. Note that the evaluation of such a protocol is out-of-scope because its goal is to determine the optimal batch size that ensures queues are stable, i.e., they do not grow to infinity or ensure packets arrive before their expiration time. Note, the said protocol only determines the size (number of packets) in each batch. Algo-PB is required to derive a minimal makespan for each batch. The evaluation of the said protocol is deferred to a future work.

4.4 Forest Construction

Algo-PB assumes a tree or routing information is given. This section shows how this can be constructed for WMNs with one or more gateways. Proposition 9 indicates the data transfer time will decrease if a node chooses to receive packets from a nearby gateway. Proposition 11 and 12 show that it is important to minimize the number of packets as well as hop count to the gateway. That is, the length of the personalized broadcast schedule will decrease if the total number of packets of the tree who has the most packets in the forest is minimized. Further, for a small Δ, say one, gateway node is able to send only one packet every two slots. Thus, the number of packets to be delivered dictates the makespan of the personalized broadcast schedule. However, for a large Δ value, i.e., each node has sufficient antennas to transmit all its packets to its descendants as well as cancel NI, the personalized broadcast time is equal to the number of hops. Recall that the weight of a node v with gateway s is defined as $w_v^s = a \times r_v + b \times dist(v, s)$, where $dist(v, s)$ is the distance (in number of hops) between node v and the gateway s and r_v is the number of packets destined for node v. According to Proposition 10, when Δ is large, the personalized broadcast makespan is determined by the number of levels of the tree. Moreover, when Δ is small, the personalized broadcast makespan is affected by the number of packets buffered at each node. To this end, for large Δ values, a is set to a small value while b needs to be large. Conversely, for small Δ values, a needs to be large and b needs to have a small value. To simplify, a and b are set to one in the following sections. Lastly, recall that given the descendants of gateway s, i.e., V_s, the weight/load of gateway s is defined as $\mathcal{W}_s = \sum_{v \in V_s} w_v^s$.

4.4.1 An ILP

The forest construction problem is first formulated precisely using an ILP. Assume a binary decision variable X_v^s is used to indicate whether node v is a descendant of gateway s. Consequently, there are $|S|$ decision variables associated with each node

v, i.e., X_v^s, where $s = 1 \ldots |S|$. Moreover, as noted before, assume that each node can only receive packets from one gateway. Thus, each node can only belong to one gateway. Let P_v^s be the shortest path from gateway s to node v, and is represented as a set of links $\{e_{(s,i)}, e_{(i,j)}, \ldots, e_{(m,v)}\}$, where for each link $e_{(i,j)}$, the endpoint or node i is the parent of node j. Note, links from gateway s to its children; e.g., link $e_{(s,i)}$, are excluded in the ILP formulation, because the gateways are always on. Define the link set $\mathcal{L}_s = \bigcup_{v \in V_s} P_v^s$. Thus, the total load of a gateway s is $\mathcal{W}_s = \sum_{v \in V} w_v^s X_v^s$. The ILP for the problem at hand is as follows,

$$\textbf{MIN} \quad MAX\{\mathcal{W}_s \mid s \in S\} \tag{4.1}$$

Subject to:

$$\sum_{v \in V_s} X_v^s = 1, \forall s \in S \tag{4.2}$$

$$X_a^s \geq X_b^s, \forall e_{a,b} \in \mathcal{L}_s, \forall s \in S \tag{4.3}$$

$$X_v^s \in \{0, 1\}, \forall v \in V - S, \forall s \in S \tag{4.4}$$

Constraint (4.2) ensures each node is connected to only one gateway. Inequality (4.3) means if a node b is a descendant of gateway s, its parent node a must be a descendant of the same gateway s. Constraint (4.4) restricts all decision variables to be binary.

In the ILP given above, each node $v \in V - S$ chooses one of the $|S|$ gateways to receive packets, thus there is $|V - S||S|$ decision variables in total. This means the number of decision variables will increase with the network size. According to Eq. (4.2), the number of constraints is equal to the number of non-gateway nodes $|V - S|$. The number of constraints acquired from Eq. (4.3) is dependent on the 'shape' of the network. In the worst case, each gateway only has one neighbor. Thus, in this case, each gateway $s \in S$, incurs $|V - S| - 1$ constraints. In total, there are $|S|(|V - S| - 1)$ constraints. According to Eq. (4.4), there are $|V - S||S|$ additional constraints. So in total, the number of constraints are $(2|S| + 1)|V - S| - |S|$. Thus,

when the network size is large, it is computationally intractable. The following section presents a greedy heuristic algorithm, called Algo-FC, to construct a routing forest.

4.4.2 Heuristic Solution

The proposed forest construction algorithm, Algo-FC, is based on the well-known Breadth-First Search (BFS) algorithm. Given a network G and $|S|$ gateways, Algo-FC first creates a tree rooted at a virtual node v' where nodes in S are the virtual node's children. Then Algo-FC creates $|S|$ sub-trees, each rooted at one gateway $s \in S$ by performing a BFS of the tree rooted at v'. After removing the virtual node v', there are $|S|$ trees, each rooted at one gateway. Algo-FC then *balances* the weight of each gateway starting with the heaviest tree. For the heaviest tree, say T^k, Algo-FC checks nodes level by level, starting from the top, to determine whether a node can be *migrated* to a different tree.

Node migration is required to balance the load between different gateways. Algo-FC *migrates* a node v and all its descendants from tree T^k to another tree T^m if a) node v has one neighbor that is associated to T^m, and b) the load of gateway m, $\mathcal{W}_m$, plus the weight of node v and all its descendants is less than $\mathcal{W}_k$ before migrating nodes. Here T^m is selected from the tree with lowest weight among all trees. Algo-FC *migrates* node v from T^k to T^m by removing the link between v and its parent on T^k and connects node v to another parent node that belongs to T^m. Note that node v relays packets for all its descendants. This means when node v is migrated to T^m, all its descendants receive packets from T^m.

Next, with the aid of Algorithm 3, Algo-FC is now described in detail. Algo-FC first creates a BFS tree rooted at the virtual node v', where gateway nodes in S are the first level nodes of the BFS tree. After performing BFS and removing the virtual node, there are $|S|$ trees, each rooted at one gateway; see Line 1. Algo-FC then calculates the load of the gateway for each tree by summing up its descendants'

weight; see Line $2-4$. Algo-FC works in rounds and uses a counter c to record the current tree in which Algo-FC is trying to balance. Initially, c is equal to one, which means Algo-FC starts at the heaviest tree; see Line 5. In each round, Algo-FC picks the c-th heaviest tree, denoted as T^k, and checks whether there is a node on T^k that can be migrated to another tree T^m. Algo-FC uses two variables, namely NF and $flag$, to indicate whether the current round is finished and whether Algo-FC successfully migrates a node in the current round, respectively; see Line $7-8$. In each round, Algo-FC checks nodes level-by-level (starting at the root's children). Algo-FC migrates a node v from T^k to T^m, if after this migration, the $\mathcal{W}_m$ is less than the $\mathcal{W}_k$; see Line $9-23$. Note, in each round, Algo-FC migrates at most one node. If Algo-FC successfully migrates a node in the current round, c is set to 1, otherwise the counter increases by one; see Line $24-28$. Algo-FC stops when no node can be migrated.

A concrete example using Figure 4.3 of Algo-FC is now presented. Figure 4.3(a) shows a network with three gateways s_1, s_2 and s_3. Assume each node has $r_v = 1$. The first step is to create a virtual node v' and perform BFS; see Line 1. This results in three trees, rooted at s_1, s_2 and s_3 respectively; see Figure 4.3(b). We use T^1, T^2 and T^3 to represent the tree rooted at s_1, s_2 and s_3 respectively. Algo-FC then calculates the weight of each tree, where $\mathcal{W}_1 = 11$, $\mathcal{W}_2 = 8$ and $\mathcal{W}_3 = 2$; see Line $2-4$. Initially c is set to one, meaning Algo-FC starts at the heaviest tree, in this example, T^1; see Line 5. In the first round, Algo-FC checks whether a node on T^1 can be migrated. Starting from the first level, i.e., s_1's children, Algo-FC ignores node A. This is because migrating A results in $\mathcal{W}_2 = 19$, which is greater than $\mathcal{W}_1 = 11$ before migration. For the same reason, Algo-FC will not migrate node F. In addition, node D and E do not have neighboring nodes associated to either tree T^2 or T^3, and thus they cannot be migrated; see Line $9-23$. We see that in this round, Algo-FC did not migrate any node. It thus increases c by one, see Line $24-28$, meaning Algo-FC will check nodes on the second heaviest tree rooted at s_2 in round two. During round two, Algo-FC first checks s_2's children. Algo-FC does

Algorithm 3: Algo-FC

Input : G, S, V, r_v for each $v \in V - S$
Output: Routing trees T^s for each $s \in S$
// Create a BFS tree from G rooted at a virtual node v' where nodes in S are the first level nodes
$\{T^1, T^2 \dots T^{|S|}\} = BFS(G, S, v')$;
for *s = 1 to $|S|$* **do**
  $\mathcal{W}_s = \sum_{v \in T^s} w_v^s$;
end
$c = 1$;
while $c \leq |S| - 1$ **do**
  // Balance the weight between different trees
  $NF = 1$;
  $flag = 1$;
  while *NF = 1* **do**
    $NF = 0$;
    $k = c_{th}$ heaviest tree;
    // L_k denotes the number of levels of the tree rooted at k
    for $i = 1$ *to* L_k **do**
      // $\mathcal{U}_v$ denotes the descendants of v, $\mathcal{N}_i$ the set of nodes at level i, and T^m is another tree that has a descendant as v's neighbor
      for *each* $v \in \mathcal{N}_i$ **do**
        if $\mathcal{W}_m + w_v^k + \sum_{d \in \mathcal{U}_v} w_d^k < \mathcal{W}_k$ **then**
          // Migrate node u from T^k to T^m
          $Migrate(v, T^k, T^m)$;
          $Update(T^k, T^m)$;
          $NF = 1$;
          $flag = 0$;
          break;
        end
      end
    end
  end
  if *flag = 0* **then**
    $c = 1$;
  else
    $c = c + 1$;
  end
end

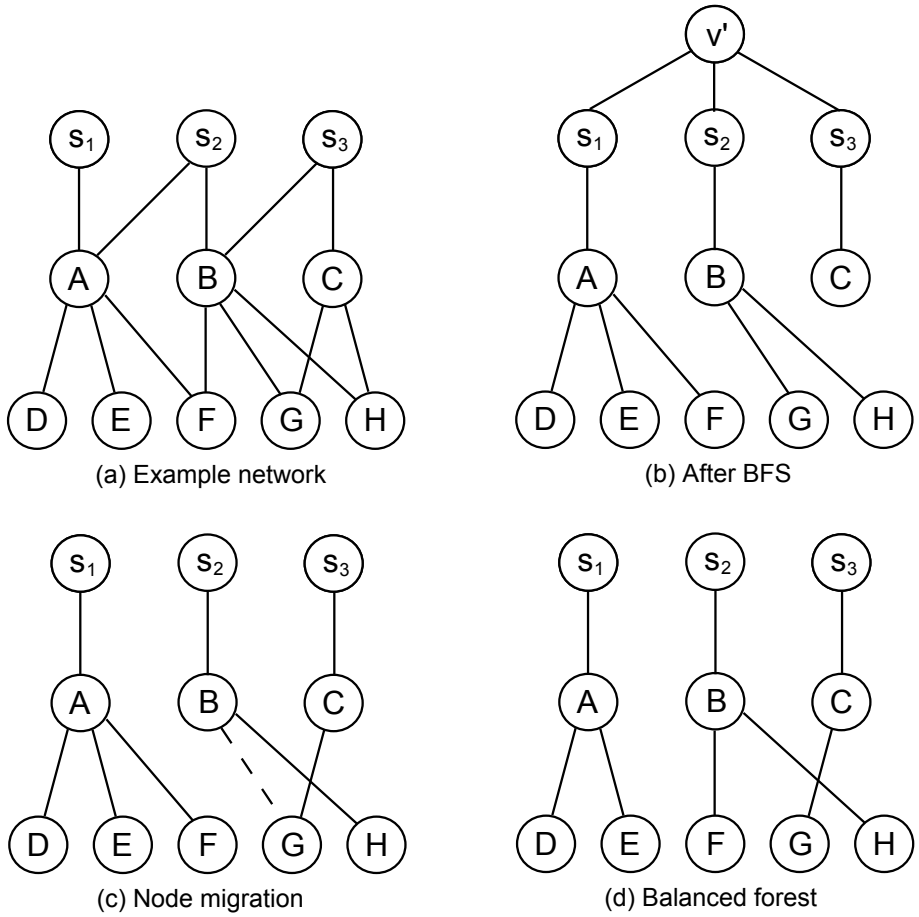


Figure 4.3: Example topology

not migrate node B because doing so results in $\mathcal{W}_3 = 10$, which is larger than $\mathcal{W}_2$ before migration. However, migrating node G to tree T^3 means $\mathcal{W}_3 = 5$, which is less than $\mathcal{W}_2 = 8$. Thus, Algo-FC migrates node G to the tree rooted at s_3 and the current round finishes; see Line $9 - 23$ and Figure 4.3(c). As Algo-FC successfully migrated one node in round two, the counter is reset to one; see Line $24 - 28$. In round three, Algo-FC has $c = 1$, $\mathcal{W}_1 = 11$, $\mathcal{W}_2 = 5$ and $\mathcal{W}_3 = 5$. Algo-FC checks whether nodes on the heaviest tree T^1 can be migrated. During round three, Algo-FC migrates node F to T^2. Again as Algo-FC successfully migrates a node in round three, *counter* is reset to 1. In the following rounds, no more nodes can be migrated, thus the forest construction result of Algo-FC is shown on Figure 4.3(d).

4.5 Analysis

This section discusses several properties of Algo-PB and Algo-FC. It also gives the computational complexity of Algo-PB and the number of migrations required for Algo-FC to create a balanced forest. Note that the proposed personalized broadcast and forest construction approach are run by the gateway node that knows the topology and the number of packets to be delivered to each node.

Proposition 13. *The computational complexity of Algo-PB on a tree T^s rooted at s is $O(|V-\{s\}|(\log|V-\{s\}|+1)+\sum_{v\in V-s} r_v)$.*

Proof. Referring to Algorithm 2, the time complexity of line 1 is $O(|V-\{s\}|\log|V-\{s\}|)$. According to lines $3-15$, during each iteration, Algo-PB either schedules one packet, see line $9-13$, or removes a node whose $r_v=0$, see line $5-7$. In total, there are $\sum_{v\in V_s} r_v$ packets scheduled, which requires $\sum_{v\in V_s} r_v$ iterations. In addition, Algo-PB removes $|V_s|$ nodes with $r_v=0$. This incurs $|V_s|$ iterations. According to line $5-7$, removing $|V_s|$ nodes has a complexity of $O(|V_s|)$. We remark that CFLU-Color has a time complexity of $O(|E|)$. In line $9-13$, scheduling $\sum_{v\in V_s} r_v$ packets has a complexity of $O(\sum_{v\in V_s} r_v \times |E|)$. Thus, in total, the computational complexity of Algo-PB is $O(|V-\{s\}|(\log|V-\{s\}|+1)+\sum_{v\in V-s} r_v \times |E|)$. □

A key computation process performed by Algo-FC is balancing trees. This involves a non-negligible number of migrations. We have the following propositions.

Proposition 14. *The BFS in Step 1 of Algo-FC sets each non-gateway node u to at least one tree T^s, for any gateway $s \in S$.*

Proof. Step 1 of Algo-FC runs BFS from a virtual gateway v' which is the parent of all gateway nodes in S. The BFS will first connect v' to all of its one-hop neighbors, i.e., all gateway nodes in S. Since Algo-FC considers only connected networks, BFS will visit each non-gateway node v and thus each node v must be a descendant of v'. In addition, each v must be a descendant of at least one gateway node $s \in S$ to be reachable from v', proving the proposition. □

Proposition 15. *For an MTR-WMN that contains $|V|$ nodes and $|S|$ gateways, Algo-FC requires at most* $1 \times \lfloor \frac{|V|-|S|}{|S|} \rfloor + 2 \times \lfloor \frac{|V|-|S|}{|S|} \rfloor + 3 \times \lfloor \frac{|V|-|S|}{|S|} \rfloor \ldots + (|S|-1) \times \lfloor \frac{|V|-|S|}{|S|} \rfloor = \frac{|S|(|S|-1)}{2} \times \lfloor \frac{|V|-|S|}{|S|} \rfloor$ *migrations to get a balanced forest.*

Proof. Start from the case with $|S| = 2$ gateways, say gateway 1 and 2. From Proposition 14, in the worst case, BFS sets all $|V| - 2$ non-gateway nodes to only one tree, say T^1, and T^2 contains only gateway 2. For this case, Algo-FC needs to migrate $\lfloor \frac{|V|-2}{2} \rfloor$ nodes from T^1 to T^2 to balance the forest. Further, in the worst case, each migration moves only one node, i.e., migrating a leaf node, and thus it requires $\lfloor \frac{|V|-2}{2} \rfloor$ migrations to get a balanced forest, showing that the proposition is correct is correct for $|S| = 2$.

Next, consider the case with $|S| = 3$ gateways, say gateway 1, 2 and 3. In the worst case, BFS sets all non-gateway nodes to only one tree, e.g., T^1, and Algo-FC needs to migrate $2 \times \lfloor \frac{|V|-3}{3} \rfloor$ nodes from T^1 to T^2 and T^3. We note that migrating $\lfloor \frac{|V|-3}{3} \rfloor$ nodes from T^1 to another tree, say T^2, requires at most $\lfloor \frac{|V|-3}{3} \rfloor$ migrations. However, in the worst case, Algo-FC may need to migrate up to $\lfloor \frac{|V|-3}{3} \rfloor$ nodes to T^2 before migrating them to T^3. Thus, migrating the nodes from T^1 to T^3 may require up to $2 \times \lfloor \frac{|V|-3}{3} \rfloor$ migrations. Therefore, in total, Algo-FC requires $\lfloor \frac{|V|-3}{3} \rfloor + 2 \times \lfloor \frac{|V|-3}{3} \rfloor = 3 \times \lfloor \frac{|V|-3}{3} \rfloor$ migrations to get a balanced forest, and thus the proposition is correct for $|S| = 3$.

Finally, consider the general case with $|S|$ gateways, say 1, 2, $\ldots s$. In the worst case, all non-gateway nodes connect to one tree, e.g., T^1 after the BFS. However, to get a balanced forest, Algo-FC needs to migrate $\lfloor \frac{|V|-|S|}{|S|} \rfloor$ nodes from T^1 to each tree. Algo-FC may need to migrate up to $\lfloor \frac{|V|-|S|}{|S|} \rfloor$ nodes to tree T^{i-1} in the worst case before migrating them to T^i, where $3 \leq i \leq |S|$. Thus, Algo-FC requires at most $1 \times \lfloor \frac{|V|-|S|}{|S|} \rfloor + 2 \times \lfloor \frac{|V|-|S|}{|S|} \rfloor + 3 \times \lfloor \frac{|V|-|S|}{|S|} \rfloor + \ldots + (|S|-1) \times \lfloor \frac{|V|-|S|}{|S|} \rfloor = \frac{|S|(|S|-1)}{2} \times \lfloor \frac{|V|-|S|}{|S|} \rfloor$ node migrations to generate a balanced forest for an MTR-WMN with $|V|$ nodes and $|S|$ gateways. □

As Proposition 15 bounds the number of migrations carried out by Algo-FC, the

following corollary holds,

Corollary 3. *Algo-FC is guaranteed to stop no later than the node migration upper bound.*

4.6 Evaluation

This section presents the evaluation of Algo-PB and Algo-FC in Matlab with the Matgraph [106] toolkit. The presented results are an average of 50 simulation runs. Each simulation run uses a different topology. Also shown in the plots is the confidence interval of 50 simulation runs, where 95% of the results are within the indicated error bar. The evaluation of Algo-PB is presented first before experiment results for Algo-FC are presented in Section 4.6.2.

4.6.1 Algo-PB

Assume all nodes are static and randomly placed on a $100m \times 100m$ square area. If two nodes are placed within the transmission range of each other, which is $25m$, they are considered to be neighbors. The interference range of each node equals its transmission range. In addition, a link can be scheduled into a slot if and only if the transmitter/receiver has sufficient antenna elements to transmit/receive and also cancel interference. The gateway node is placed at the center of the square area. This thesis compares the personalized broadcast schedule length generated by Algo-PB with the theoretical upper and lower bound derived in Section 4.2 as well as the link scheduler proposed in [26]. To ensure a fair comparison, Bermond et al.'s algorithm [26] is modified to include MTR capability; the revised algorithm is called *ScheTree*,. That is, in each time slot, a node v is now able to transmit up to Δ packets. In addition, the routing tree is generated by performing a BFS at the gateway. The plotted results are the schedule length of all algorithms and compute the theoretical upper and lower bound of each topology.

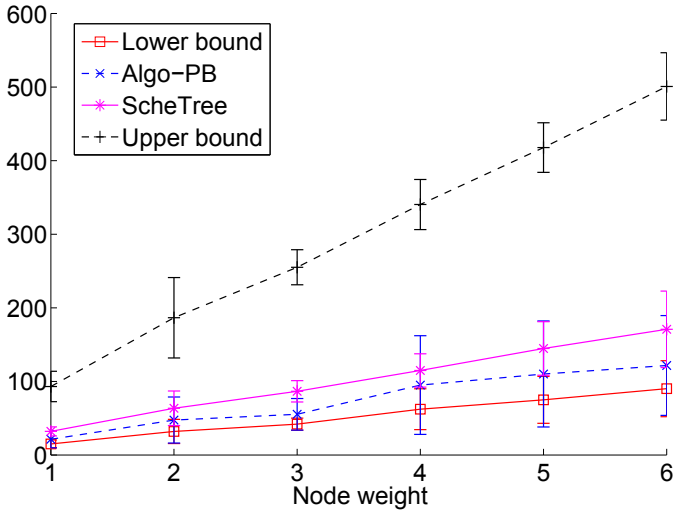


Figure 4.4: Schedule length under different node demand

4.6.1.1 Node Demand

The first result concerns different node demands; i.e., r_v for each node v. This is important because different node demands lead to different personalized broadcast schedule length. The network size is fixed to 30 and each node has $\Delta = 3$. There are six groups of experiments where the demand of each node varies from one to six. Note, all nodes are assumed to have the same demand in each simulation group, i.e., in the first group, the weight of each node is one. From Figure 4.4, the difference between Algo-PB and the theoretical lower bound is within 34.5%. Algo-PB outperforms ScheTree by generating superframe lengths that are up to 33.3% shorter. This is because Algo-PB always preferentially schedules packets to the node farthest from the gateway and does not switch to another node until a node is fully serviced. Thus, Algo-PB makes the best use of the spacial multiplexing capability of nodes to produce shorter schedules.

4.6.1.2 Node Density

Assume each node v has four antennas. There are 9 groups of experiments with network size varying from 20 to 60, with a step size of 5. The total number of packets to be delivered is set to 300 and these packets are evenly delivered to each

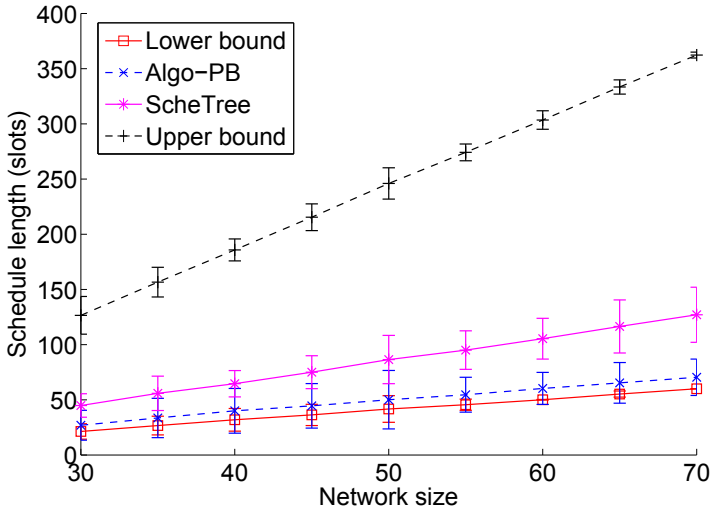


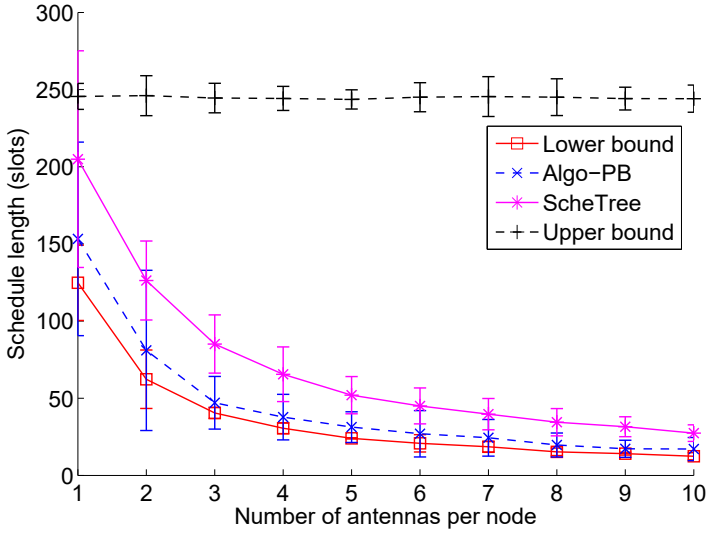
Figure 4.5: Schedule lengths under different network sizes

node. Figure 4.5 shows the resulting schedule length. The schedule length increases with network size as expected. This is because more nodes result in a routing tree with more levels. This agrees with the analysis in Section 4.2. Simulation results show that Algo-PB outperforms ScheTree by producing up to 44.6% shorter schedules. The difference between Algo-PB and the theoretical lower bound is at most 20.9%.

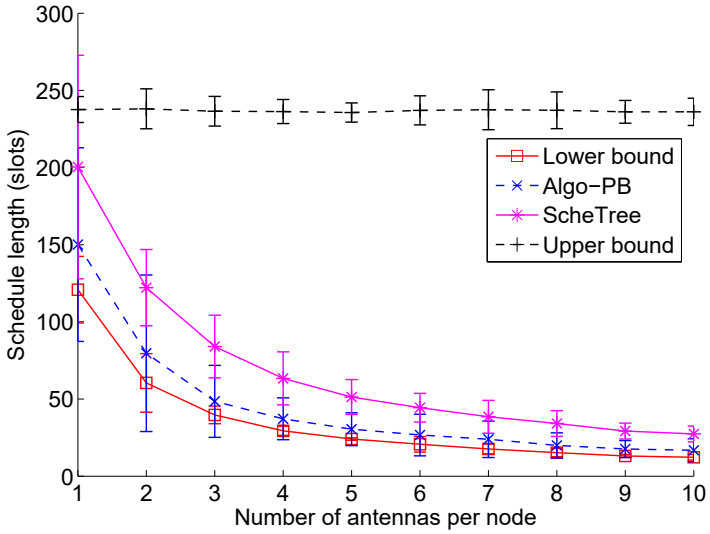
4.6.1.3 Number of Antennas

Lastly, this section focuses on the number of antennas at each node. The network size is set to 40 nodes. Figure 4.6(a) shows the schedule length when the weight of each node is three. On the other hand, in Figure 4.6(b), the weight of each node is randomly chosen from the range $[0, 5]$. When the number of antennas increases, the generated schedule length decreases. This is because nodes are able to receive/forward more packets in each slot with increasing number of antennas. Compared with ScheTree, Algo-PB generates at most 44.7% shorter schedule lengths when all nodes have a weight of three, and 42.3% shorter schedule lengths when each node has a random weight. The difference between Algo-PB and the theoretical lower bound is at most 26.5%. Algo-PB has better performance because

it schedules the farthest unscheduled packet in each round and makes the best use of spatial multiplexing to produce shorter schedules.



(a)



(b)

Figure 4.6: Schedule length versus Δ_u

4.6.2 Forest Construction

The next set of results concern Algo-FC. Assume all nodes are stationary and randomly connected. For each node, the number of requests, i.e., r_v, is randomly chosen from the range $[0, 5]$. There are three sets of evaluation. First, the number of nodes

varies from 10 to 70. The number of gateways is set to four and node degree is also set to four. Second, the conducted experiments considers 35 nodes and four gateways. The degree of each node varies from three to six. Third, The network size is fixed to 35, node degree is set to four. However, the number of gateways varies from two to five.

Algo-FC and ILP are compared with two other solutions, breadth-first searching (BFS) and weight focus routing (WFR). Briefly, BFS and WFR work as follows:

- *BFS.* First, a virtual node is created and is connected to all gateways. Then a BFS is performed from the virtual node. Finally, the virtual node is removed. This process results in multiple trees where each node is associated with the nearest gateway.

- *WFR.* It repeatedly picks the node v with the largest r_v that is not connected to any tree. It then finds a path p from node v to the nearest gateway s. WFR connects all nodes along path p to gateway s. Note that if a node n on path p is already connected to another tree rooted at gateway k, WFR connects nodes from v to n to gateway k and ignores other nodes.

In each experiment, the following metrics are recorded:

- *Maximum gateway load.* This is the load of the heaviest gateway. It is calculated by summing the weight of all nodes served by a gateway.

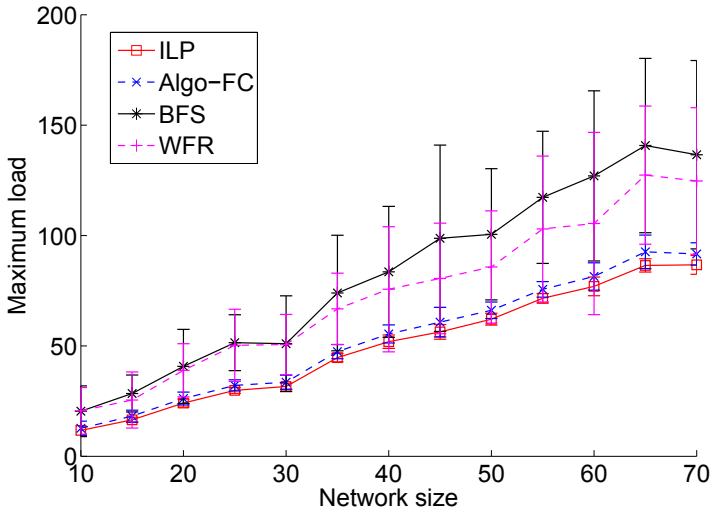
- *Computation time.* This is the time consumed to generate the forest, measured in seconds. Matlab's optimization toolbox is used to solve the ILP. The experiments in this section are conducted on a computer with an Intel Core i5 processor with 8 GB RAM. As noted in Section 4.1, the forest construction algorithm is run at gateway node.

4.6.2.1 Network Size

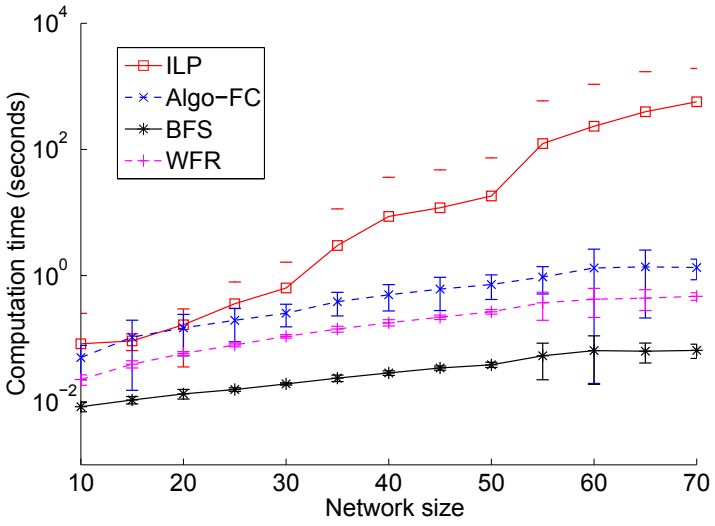
In this set of experiments, the network size is varied from 10 to 70. The number of gateways is set to four and the node degree four. Results in Figure 4.7 show that with increasing network size, the load of the heaviest gateway increases and the computation time increases. Specifically, in Figure 4.7(a) we see that Algo-FC outperforms BFS and WFR by at least 60.6% and 62.2%, respectively. This is because BFS only considers the shortest path and WFR only focuses on the heaviest nodes. Also, BFS and WFR do not consider the load of gateways. Compared to the ILP, Algo-FC results in the heaviest gateway having at most 8.6% higher load than the ILP. In terms of computation time, BFS has the best performance. As shown in Figure 4.7(b), when the network size increases, the computation time for ILP increases exponentially as expected. When the network size is 70, it takes 570.1 seconds on average for the ILP solver to get the optimal solution. However, Algo-FC only requires 1.34 seconds to compute a solution when the network size is 70. Algo-FC balances the load between gateways, unlike BFS and WFR, and thus the computation time for Algo-FC is higher than the latter two algorithms.

4.6.2.2 Node Density

Increasing node density results in a higher node degree. Thus, in this group of experiments, the network size is fixed at 35 and the number of gateways is set to four. Figure 4.8(a) shows that higher node densities tend to decrease the load of the heaviest gateway. Compared to ILP, Algo-FC results in the heaviest gateway having at most 9.1% higher load. Algo-FC outperforms BFS and WFR by at least 43.1% and 43.7%, respectively as BFS and WFR do not seek a balanced forest. In terms of computation time, Figure 4.8(b) shows that ILP requires the longest time as expected. When each node has six neighbors, ILP takes 58.8 seconds. However, Algo-FC only needs 0.36 second under the same condition.

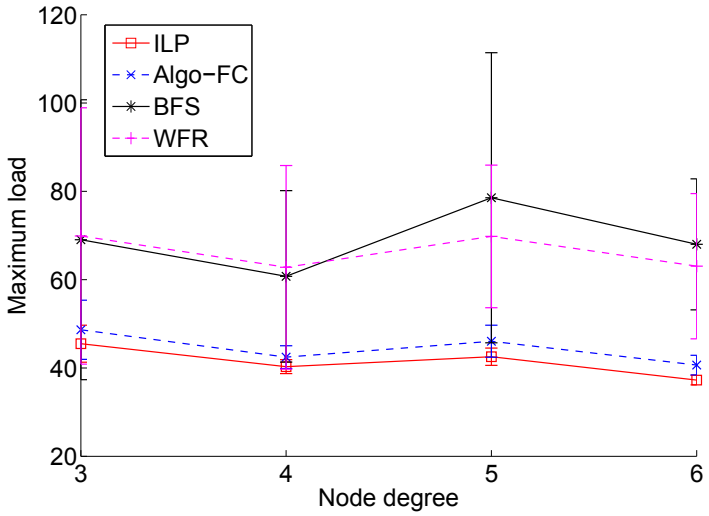


(a)

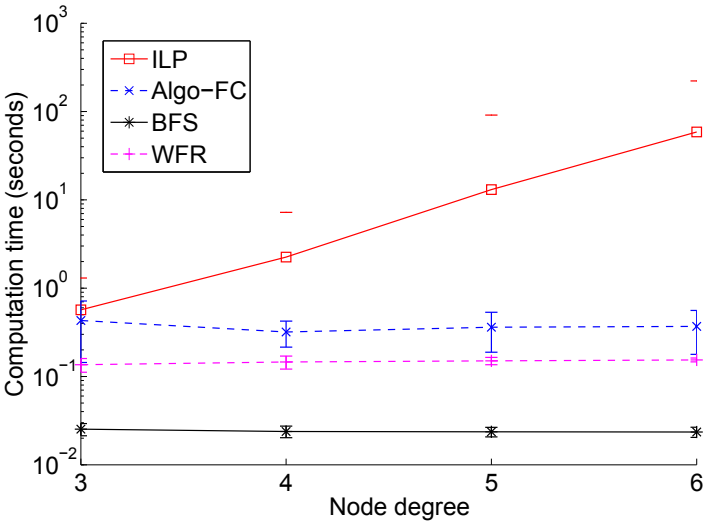


(b)

Figure 4.7: Performance under different network sizes



(a)



(b)

Figure 4.8: Performance under different node densities

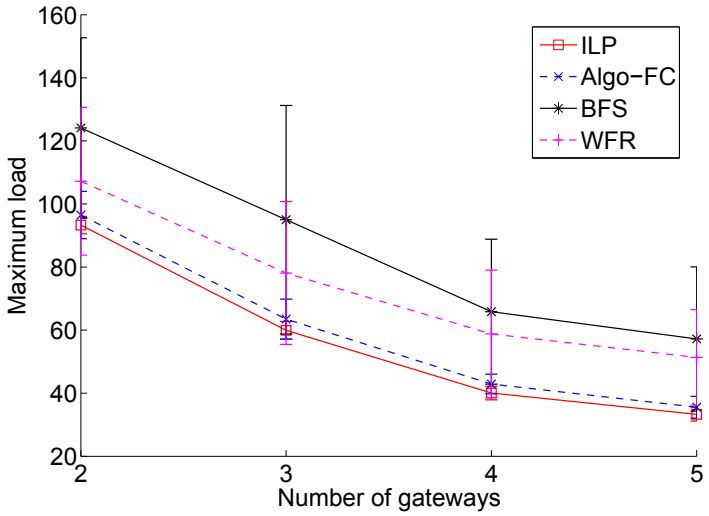
4.6.2.3 Number of Gateways

Finally, this group of experiments test the performance of the four approaches under different number of gateways. Figure 4.9(a) indicates that the load of the heaviest gateway decreases with increasing number of gateways. This is because adding more gateway nodes results in more sources that are transmitting packets. Algo-FC generates at most 6.7% higher load at the heaviest gateway as compared to ILP and outperforms BFS and WFR by 60.1% and 44.4% respectively. Figure 4.9(b) shows that the computation time of Algo-FC is 0.48 seconds on average when there are five gateways. However, ILP requires 14.6 seconds on average when there are five gateways.

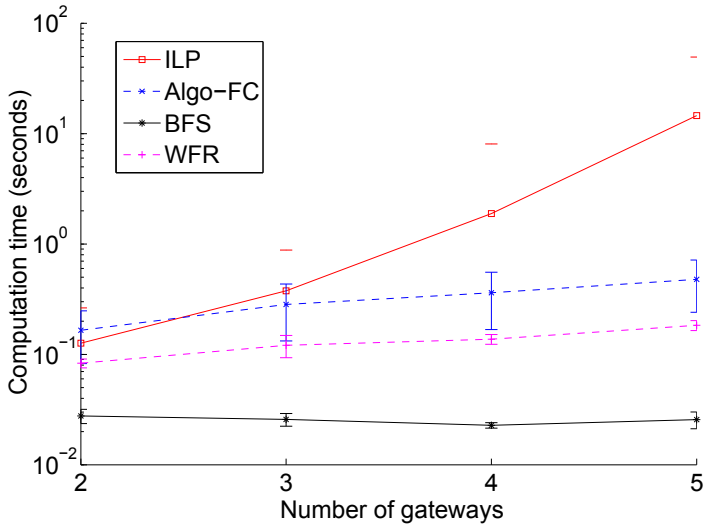
4.7 Conclusion

This chapter has addressed the personalized broadcast problem and the forest construction problem. It also presented bounds for the personalized broadcast schedule in tree-based MTR-WMNs and proposed a novel link scheduler to generate a personalized broadcast schedule with minimal makespan. This chapter also formulated the forest construction problem using an ILP and proposed a heuristic algorithm called Algo-FC to generate the routing forest. Through comprehensive experiments, Algo-PB is shown to outperform current algorithms and is within 34.5% of the theoretical lower bound. Also, compared to the optimal solution generated by the ILP, the weight of the heaviest gateway produced by Algo-FC is at most 9.1% longer.

A key limitation of Algo-PB is that it considers packets flow in a single direction only. However, in practice, packets flow in both directions. Henceforth, the next chapter addresses the problem of scheduling packets to/from a gateway by taking advantage of full-duplex nodes.



(a)



(b)

Figure 4.9: Performance under different number of gateways

Chapter 5

On Uplink and Downlink Packet Scheduling in Full-Duplex Wireless Mesh Networks

To date, only a handful of TDMA link schedulers exist for full-duplex wireless networks [98] [101]; both of which only consider single-hop packet transmissions. Further, there is no existing CSMA or TDMA link schedulers that consider delivering both uplink and downlink packet over multiple hops in minimum time. Thus, this chapter focuses on generating a minimal TDMA schedule to transfer both uplink and downlink packets for each node in a WMN. Specifically, it presents two link schedulers, namely UDMAC and Algo-UD, that generate a link schedule in a path-by-path and slot-by-slot manner, respectively. Moreover, both schedulers consider suppressing neighboring interference.

The remainder of this chapter is structured as follows. Section 5.1 presents the network model and describes the uplink and downlink packet scheduling problem. Section 5.2 gives the theoretical upper and lower bound of the uplink and downlink

packet scheduling schedule. UDMAC and Algo-UD are outlined in Section 5.3. Experimental results are presented and discussed in Section 5.4. Section 5.5 concludes the chapter.

5.1 Preliminaries

Consider a MU-MIMO based single channel full-duplex WMN modeled as a directed graph $G(V, E)$, where V is the set of static nodes and E contains all directed links $e_{v,u}$ from node $v \in V$ to node $u \in V$. Each node v has a transmission range R_t, and if node v and u are within range of each other, then links $e_{v,u}$ and $e_{u,v}$ exist. The interference range of node i, R_i is assumed to be equal to its transmission range; i.e., $R_i = R_t$. In addition, nodes have the channel state information (CSI) to their neighbors and are located in a rich scattering environment [108]. Each node $v \in V$ is identified by a unique integer ID. Let δ_v be the degree or number of neighbors of node v; the in-degree of each node v is equal to its out-degree. In addition, nodes are able to transmit and receive *multiple* packets to/from a neighbor concurrently.

Each node $v \in V$ is equipped with $\Delta_v^{\pm}$ antenna elements or DoFs that can be used for either transmission/reception or interference cancellation. Let Δ_v^{+}, Δ_v^{-}, Δ_v^{sc} and Δ_v^{ic} be the number of DoFs that node v uses for transmissions, receptions and self interference cancellation (SIC) and neighboring interference cancellation (NIC), respectively. According to [28], to enable full-duplex communication, for each transmission, a node must dedicate one DoF for each incoming transmission to null self-interference. Specifically, if $\Delta_v^{+} > 0$ and $\Delta_v^{-} > 0$, then

Constraint 4. $\Delta_v^{sc} = \Delta_v^{-}$.

Also, a node needs to use part of its DoFs to suppress neighboring interference (NI) caused to/from neighbors. Specifically, each NI link is canceled at only one end of an interfering link. Suppose there is a NI link between node v and its neighboring node u. If node v is responsible for canceling such NI, it needs to use Δ_u^{-} DoFs.

On the other hand, if NIC is the responsibility of node u, then it needs to use Δ_v^+ DoFs. Let $\mathcal{I}_v$ be the set of neighbors in which node v needs to suppress NI. Further, $\mathcal{I}_v^+ \subset \mathcal{I}_v$ contains v's neighbors that are receiving while $\mathcal{I}_v^- \subset \mathcal{I}_v$ denotes the set of transmitting nodes that cause NI to v. We then have

Constraint 5. $\Delta_v^{ic} = \sum_{u \in \mathcal{I}_v^-} \Delta_u^+ + \sum_{u \in \mathcal{I}_v^+} \Delta_u^-$

The total DoFs that a node uses for data transmission, reception, SIC and NIC must not exceed its total available DoFs. That is,

Constraint 6. $\Delta_v^+ + \Delta_v^- + \Delta_v^{sc} + \Delta_v^{ic} \leq \Delta_v^{\pm}$.

Let $s \in V$ be the gateway, and V_s contains all non-gateway nodes. Each node $v \in V_s$ has $u_v \geq 0$ packets to be delivered to s, and $d_v \geq 0$ to be received from gateway s. Let T be the spanning tree of G rooted at s, and T_i be the sub-tree of T rooted at the i-th child of s. Denote $V_v \subset V$ as the set of descendants of node v. For an arbitrary topology with a gateway s, all sub-trees of T are established by performing a breadth-first search (BFS) from the gateway s. Time is slotted, as facilitated by GPS, and each packet transmission consumes one slot.

The uplink and downlink packet scheduling problem is formally defined as follows:

Definition 5. For a given WMN G(V, E), derive a collision-free link schedule that transfers all u_v data packets from each node $v \in V - s$ to gateway s and d_v data packets from the gateway s to their perspective destination $v \in V_s$, in minimal time subject to constraint 4, 5 and 6.

Note that the NP-complete personalized broadcast and data gathering problems [26] are special instances of the uplink and downlink packet scheduling problem. Specifically, if u_v for each node $v \in V_s$ is set to zero, then we have the personalized broadcast problem. Alternatively, by setting $d_v = 0$ for $v \in V_s$, we have the data gathering problem.

5.2 Upper and Lower Bound

This section gives the theoretical schedule upper and lower bound for an arbitrary tree T rooted at s. From hereon, the term 'UD-Schedule' denotes the uplink and downlink packets schedule.

The next proposition upper bounds the UD-Schedule. Let L be the number of levels of tree T. Denote U_l and D_l to be the total number of uplink packets and downlink packets at level l of T respectively, for $1 \leq l \leq L$. Let V_l be a node set containing all nodes at level l. Denote $\mathcal{T}_U$ and $\mathcal{T}_D$ to be the maximum number of slots required to schedule all uplink and downlink packets for sub-tree T_i respectively. The UD-Schedule upper bound of tree T is denoted as $\mathcal{T}$.

Proposition 16. *The schedule to transmit all downlink packets is upper bounded by* $\mathcal{T}_D = \sum_{l=1}^{L} (l + 2(D_l - 1))$ *slots.*

Proof. The first downlink packet to arrive at level l of T requires l slots. In the worst case, a transmitting node at level l interferes with all nodes at level $l-1$, l and $l+1$. Thus the second packet to level l arrives at the $(l+2)$-th slot, the third packet arriving at the $(l+4)$-th slot and so forth. Consequently, transferring D_l packets to level l requires up to $l+2(D_l-1)$ slots. In the worst case, gateway s starts to transmit downlink packets located at level l after it has finished transferring all downlink packets designated for nodes at level $l+1$. Thus, for a tree T, it will take no more than $\mathcal{T}_D = \sum_{l=1}^{L} (l + 2(D_l - 1))$ slots to deliver all packets. □

As for the upper bound for uplink packets, we have,

Corollary 4. *It takes up to* $\mathcal{T}_U = \sum_{l=1}^{L} (l + 2(U_l - 1))$ *slots to transmit all uplink packets from nodes.*

Proof. The first uplink packet from level l of T requires l slots to get to the gateway, meaning U_l uplink packets require no more than $l + 2(U_l - 1)$ slots. In the worst case, nodes at level l start to transmit uplink packets after all uplink packets from

level $l+1$ arrive at the gateway. This means the number of slots required to upload all packets for T will not be longer than $\mathcal{T}_U = \sum_{l=1}^{L} (l + 2(U_l - 1))$ slots. □

In the worst case, the gateway starts transmitting downlink packets when it has received all uplink packets, which implies the following upper bound.

Corollary 5. *The UD-Schedule for tree T is upper bounded by $\mathcal{T} = \mathcal{T}_U + \mathcal{T}_D$.*

Next, this section derives the lower bound of UD-Schedule. Let P_v^+ and P_v^- be the total number of packets that node v has to transmit and receive, respectively. In Figure 5.1, suppose each node v has $u_v = d_v = 1$. Thus node A will receive two uplink packets and transmit three uplink packets. Node A will also receive three downlink packets, two of which are transmitted to its descendants. Thus, in total, we have $P_A^+ = 5$ and $P_A^- = 5$.

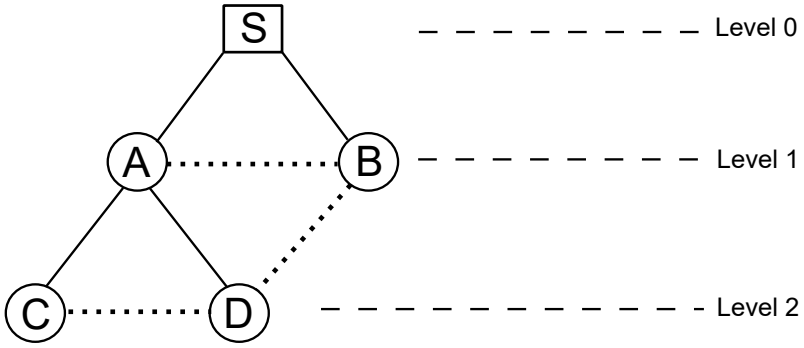


Figure 5.1: A tree topology with interfering link (dotted lines).

Proposition 17. *The lower bound of UD-Schedule without NI for a tree T is $MAX\{MAX(\lceil (P_i^+ + 2P_i^-)/(\Delta_i^{\pm}) \rceil), MAX(\lceil (P_j^+)/(\delta_j) \rceil), MAX(\lceil (P_k^-)/(\delta_k) \rceil\}$, for any $i, j, k \in V$.*

Proof. One transmit antenna is required to null self-interference. Hence, a node i that is transmitting and receiving P_i^+ and P_i^- packets, respectively, requires $2P_i^-$ antennas. This means each node $i \in V$ requires $\lceil \frac{P_i^+ + 2P_i^-}{\Delta_i^{\pm}} \rceil$ slots to finish all transmissions and receptions. Further, for each of its neighbors, any node j is only able to transmit and receive one packet. Thus, transmitting P_j^+ packets require at least

$\lceil \frac{P_j^+}{\delta_j} \rceil$ slots and receiving P_k^- packets require at least $\lceil \frac{P_k^-}{\delta_k} \rceil$ slots. Therefore, the UD-Schedule length will be no less than the stated lower bound for any $i, j, k \in V$. □

The foregone proposition serves as the best possible lower bound because nodes are assumed to not cause NI to other nodes. A bound that considers NI remains an open question.

5.3 Solution

The details of UDMAC and Algo-UD are now presented. Both schedulers are centralized solutions and run by a gateway that knows the topology and number of uplink and downlink packets. With regards to the number of uplink packets, nodes can piggyback this information in their uplink packets. In addition, both UDMAC and Algo-UD can be used with any routing protocols. The only requirement is that nodes have a route to the gateway and vice-versa. The difference between UDMAC and Algo-UD are as follows. First, UDMAC schedules packets in a path-by-path manner while Algo-UD generates the link schedule in a slot-by-slot manner while maximizing the number of packets scheduled in each slot. Second, UDMAC assumes NI is canceled at the transmitter node while Algo-UD uses a node ordering rule to give each node an order and each NI link is canceled by the end node with a higher order.

5.3.1 UDMAC

Each path p is a vector containing the ID of all nodes along the path. Denote $p(1)$ to be the source node of path p and $p(|p|)$ is the destination node. Further, $e_{p(i),p(i+1)}$ is a link on path p from $p(i)$ to $p(i+1)$ for $1 \leq i \leq |p| - 1$. The key idea is to always schedule one packet to/from the node farthest from the gateway that has unscheduled packets. If the farthest node has more uplink packets than downlink packets, UDMAC schedules one uplink packet. Otherwise, it schedules one downlink

packet. Let $\mathcal{V}$ be a vector containing nodes in decreasing hop length order from the gateway s, and $\mathcal{V}(1)$ is the farthest node from s. Denote F to be the final schedule and t_{max} to be the final schedule length. The number of antennas used by a node in each time slot is recorded in a $|V| \times t_{max}$ matrix called $\mathcal{A}$. For example, if a node with ID 6 uses one antenna for transmission, two antennas for reception, two antennas for SIC and two antennas for NIC in slot 3, thus $\mathcal{A}(6,3) = 1+2+2+2 = 7$.

A key part of UDMAC is slot assignment. This is carried out using a function called Collision Free Full-Duplex Coloring or CFFD-Color(); a modified version of the Conflict Free Coloring (CF-Coloring) algorithm of [26]. Given a path p, CFFD-Color() assigns an increasing positive integer or color to each link starting from the source node subject to constraint 4, 5 and 6. In addition, the color assigned to each link along path p must be strictly increasing starting from the source node. This ensures links are activated consecutively. For each path p, CFFD-Color() generates its link schedule, Sched(p), represented by a tuple $<(e_{1,2}, c_\alpha), (e_{2,3}, c_\beta), \ldots, (e_{i,j}, c_\gamma)>$, where $e_{i,j}$ denotes the link from i to j and c_γ is the color of link $e_{i,j}$. Note, links assigned with color c_γ will be activated in the c_γ-th slot.

The pseudo-code of CFFD-Color() is shown in Algorithm 4. Consider the topology shown in Figure 5.1; it has paths $p_1 = S \rightarrow A \rightarrow C$, $p_2 = A \rightarrow S$ and $p_3 = S \rightarrow B$. Assume each node has $\Delta_v^{\pm} = 3$ antennas, and the node ID of A, B, C and S is $1, 2, 3$ and 5, respectively. We also have paths $p_1 = [5, 1, 3]$, $p_2 = [1, 5]$ and $p_3 = [5, 2]$. Initially all the values in $\mathcal{A}$ are zero. We start with path p_1 because it is the longest. Initially c is set to one; see Line 1 of Algorithm 2. CFFD-Color() starts by assigning a color to the first link of path p_1, which is $e_{5,1}$. Before assigning a color c to $e_{5,1}$, CFFD-Color() first checks, using DoFCheck(), whether assigning color c to $e_{5,1}$ satisfies constraints 4, 5 and 6. If a constraint is not satisfied, DoFCheck() returns '1', which means color c is ineligible for link $e_{5,1}$. Otherwise, DoFCheck() returns '0'; see Line 3-5 of Algorithm 2. Referring to Figure 5.1, color $c = 1$ can be assigned to link $e_{5,1}$ as it satisfies constraints 4, 5 and 6. Then CFFD-Color updates $Sched(p)$ and $\mathcal{A}$; see Line 6-7. Color c is also incremented by one; see Line 9. Next,

assigning color $c = 2$ to $e_{1,3}$ satisfies constraint 4 to 6, thus color 2 is assigned to link $e_{1,3}$; see Line 2-9. At this point, each link in p_1 has a color and we have Sched(p_1) = $<(e_{5,1}, 1), (e_{1,3}, 2)>$. Next, consider $p_2 = [1, 5]$. Initially c is set to one. Assigning $c = 1$ to link $e_{1,5}$ satisfies constraints 4 to 6, thus link $e_{1,5}$ is associated with color 1. We then have Sched(p_2) equals to $<(e_{1,5}, 1)>$. Finally, we consider path p_3. Assigning $c = 1$ to link $e_{2,5}$ does not satisfy constraint 6. This is because link $e_{1,5}$ and $e_{5,1}$ are both assigned color $c = 1$. Thus if we assign $c = 1$ to link $e_{2,5}$, node A needs to use one antenna to transmit to S, one antenna to receive from S, one antenna to cancel self-interference and another antenna to cancel NI for node B caused by $e_{1,5}$; in total, four antennas are required, which exceeds $\Delta_A^{\pm} = 3$.Assigning $c = 2$ to link $e_{2,5}$ satisfies constraints 4, 5 and 6. Thus we have Sched(p_3) = $<(e_{2,5}, 2)>$.

Algorithm 4: CFFD-Color()

Input : p, $\mathcal{A}$
Output: Schedule $Sched(p)$, Updated $\mathcal{A}$

```
1 c = 1, Sched(p) = ∅;
2 for i ← 1 to |p| − 1 do
3 |   while DoFCheck(p, i, c, A) do
4 |   |   c = c + 1 ;
5 |   end
6 |   Sched(p) = Sched(p) ∪ (e_{p(i),p(i+1)}, c);
7 |   Upd(p, i, c, A);
8 |   c = c + 1;
9 end
```

The details of UDMAC are now presented with the aid of Algorithm 4 and 5. UDMAC first sorts all nodes in decreasing hop length order from the gateway and stores them in the vector $\mathcal{V}$. In each iteration, UDMAC first picks the farthest node v from the gateway s and checks whether all its uplink and downlink packets are scheduled; i.e., $u_v + d_v = 0$. If so, UDMAC removes v from $\mathcal{V}$ and moves to the next iteration; see Line 4-6. Otherwise, it checks whether node v has more uplink than downlink packets. In the former, UDMAC obtains a path p from v to s. Otherwise, it selects a p that originates from the gateway s to v. UDMAC then employs CFFD-Color() to schedule all links in p, i.e., assign one slot to each of its links; see Line 14.

Algorithm 5: UDMAC

Input : V, d_v, u_v, $for\ each\ v \in V_s$
Output: $makespan(F)$, F

1 $\mathcal{V} = Sort(V)$, $F = \emptyset$;
2 **while** $\mathcal{V} \neq \emptyset$ **do**
3 $\quad v = \mathcal{V}(1)$ `// Get the first node in` $\mathcal{V}$
4 $\quad$ **if** $u_v + d_v = 0$ **then**
5 $\quad\quad \mathcal{V} = \mathcal{V} - v$;
6 $\quad$ **else**
7 $\quad\quad$ **if** $u_v > d_v$ **then**
8 $\quad\quad\quad p = path(v \rightarrow s)$;
9 $\quad\quad\quad u_v = u_v - 1$;
10 $\quad\quad$ **else**
11 $\quad\quad\quad p = path(s \rightarrow v)$;
12 $\quad\quad\quad d_v = d_v - 1$;
13 $\quad\quad$ **end**
14 $\quad\quad (Sched(p), \mathcal{A}) =$ CFFD-Color$(p, \mathcal{A})$;
15 $\quad\quad F = F \cup Sched(p)$;
16 $\quad$ **end**
17 **end**
18 return$(makespan(F), F)$;

The schedule derived by CFFD-Color() is stored in $Sched(p)$ and is added to the final schedule F; see Line 15. Lastly, UDMAC returns the schedule length and F. At this point, the gateway informs nodes their transmission slots and start time by piggybacking this information in their downlink packets. If there are no downlink packets, then a dummy packet can be created before the schedule is computed. Note this information requires modest overheads. Nodes only require their assigned slot IDs, and for queue information, hence, 8 bytes will be sufficient.

Lastly, UDMAC has a run time complexity of $\mathcal{O}(|E| \times \sum_{v \in V_s} (u_v + d_v))$. This is because CFFD-Color() has a run time complexity of $\mathcal{O}(|E|)$. For each uplink and downlink packet, UDMAC needs to run CFFD-Color(); this function is called a maximum $\sum_{v \in V_s} (u_v + d_v)$ times.

5.3.2 Algo-UD

Different from UDMAC, Algo-UD generates the UD-Schedule in a slot-by-slot manner. Briefly, Algo-UD works as follows. In each slot, Algo-UD sorts all nodes in decreasing order according to the number of packets buffered at each node. Then Algo-UD starts with the node that has the most buffered packets, say node v. It then schedules as many of these packets from node v as possible; this is subject to constraints 4, 5 and 6. After draining all of node v's packets, Algo-UD moves to the next node with the highest number of packets. Algo-UD continues this process until it empties the buffer of all nodes. Unlike UDMAC, Algo-UD uses node ordering to determine which end node of an NI link needs to suppress such interference. Specifically, consider the following example. Assume we have a sorted list containing three nodes: <A, B, C >. Then node B is responsible for suppressing all NI to/from node A. Further, the NI between node B and C must be suppressed by node C.

Before presenting the details of Algo-UD using Algorithm 6, a few more notations are in order. Let F denote the final schedule generated by Algo-UD; that is, the schedule specifies the set of links activated in each slot. The set of links activated in slot t is denoted as F_t. Algo-UD uses a vector $\mathbf{a} \in \mathbb{N}^{|V|}$ to record the number of DoFs used by each node. Specifically, entry $\mathbf{a}_i$ records the total number of antennas node i uses for transmission, reception and IC. For example, if the node with ID 6 uses one DoF for transmission, two DoFs for reception, two DoFs for SIC and one antenna for NIC, then we have $\mathbf{a}_6 = 1+2+2+1 = 6$. Let $\mathbf{p}$ be a vector that records the path of one uplink/downlink packet. Specifically, $\mathbf{p}$ contains the ID of all nodes along the path of a packet from the packet's current location to its destination. As before, the routing tree is constructed via a BFS from the gateway. Thus the path $\mathbf{p}$ is the shortest path from a packet's current location to its destination along the edges of the routing tree. Further, let $\mathbf{p}(1)$ to be the current location of the packet recorded in $\mathbf{p}$ and $\mathbf{p}(|\mathbf{p}|)$ to be the destination. Further, $e_{\mathbf{p}(i),\mathbf{p}(i+1)}$ is a link on path $\mathbf{p}$ from $\mathbf{p}(i)$ to $\mathbf{p}(i+1)$ where $1 \leq i \leq |\mathbf{p}| - 1$. Denote $\mathbf{P}$ to be a tuple containing

all path $\mathbf{p}$. The variable p_i is used to denote the i-th packet recorded in $\mathbf{P}$ and $\mathbf{p}_i$ to denote the path of the i-th packet. Referring to Figure 1.7, assume each node 1, 2 and 3 has one uplink packet and one downlink packet. We then have $\mathbf{P}$ =<[1 4], [4 1], [2 4], [4 2], [3 1 4], [4 1 3] >.

Algorithm 6: Algo-UD

Input : V, d_v, u_v, $for\ each\ v \in V_s$
Output: F, t

1 $\mathbf{P} = \emptyset$, $F = \emptyset$, $\mathbf{a}_i = 0$, where $1 \leq i \leq |V|$;
2 **for** $i \leftarrow 1$ **to** $\sum_{v \in V}(u_v + d_v)$ **do**
3 | $Update(\mathbf{P})$;
4 **end**
5 $t = 0$;
6 **while** $\mathbf{P} \neq \emptyset$ **do**
7 | $t = t + 1$;
8 | $F_t = \emptyset$;
9 | $\mathbf{v} \leftarrow sort(V)$;
10 | $\mathbf{P} \leftarrow sort(\mathbf{P})$;
11 | **for** $v \leftarrow 1$ **to** $|\mathbf{v}|$ **do**
12 | | $Update(o_v)$;
13 | **end**
14 | **for** $i \leftarrow 1$ **to** $|\mathbf{v}|$ **do**
15 | | **for** $j \leftarrow 1$ **to** $size(\mathbf{P})$ **do**
16 | | | **if** $\mathbf{p}_j(1) \neq \mathbf{v}_i$ **then**
17 | | | | continue;
18 | | | **end**
19 | | | **if** $DoFCheck(\mathbf{p}_j(1), \mathbf{p}_j(2), \mathbf{a}, o_{\mathbf{p}_j(1)}, o_{\mathbf{p}_j(2)}) = 1$ **then**
20 | | | | continue;
21 | | | **end**
22 | | | $F_t = F_t \cup e_{\mathbf{p}_j(1),\mathbf{p}_j(2)}$;
23 | | | $Update(\mathbf{a})$;
24 | | **end**
25 | **end**
26 | $F = F \cup F_t$;
27 | $Update(\mathbf{P})$;
28 **end**
29 $return(F, t)$;

Algo-UD first sets $\mathbf{P}$, F to empty and all entries in $\mathbf{a}$ to zero; see Line 1. Next, Algo-UD records the path of all uplink and downlink packets in tuple $\mathbf{P}$; see line $2-4$. Algo-UD keeps a counter t to track the slot number; see line 5. To generate the schedule for slot t, Algo-UD first sorts all nodes in V in decreasing order according

to the number of packets buffered at each node, and stores sorted nodes in the vector $\mathbf{v}$; see Line 9. Algo-UD also sorts the path of all packets recorded in $\mathbf{P}$ in decreasing distance order; see Line 10. Algo-UD then associates each node v with order o_v; a node with more buffered packets has a higher order. Ties are broken using node ID; see Line $11-13$. Next, Algo-UD schedules packets for each node $v \in \mathbf{v}$; it starts with the node that has the highest order. To schedule packets buffered at node v, Algo-UD checks, using DoFCheck(), each packet at node v whether it can be scheduled in slot t. Algo-UD starts with the packet farthest from its destination. If scheduling the j-th packet buffered at v in slot t does not satisfy either Constraints 4, 5 or 6, DoFCheck() returns 'true' and moves to the $j+1$-th packet. Otherwise, DoFCheck() returns 'false' and link $e_{\mathbf{p}_j(1),\mathbf{p}_j(2)}$ is added to F_t; see Line $14-25$. After Algo-UD schedules all possible packets for slot t, Algo-UD adds F_t to the final schedule F in Line 26. Algo-UD also updates the path of each scheduled packet. For instance, the j-th packet recorded in $\mathbf{p}_j = [4\ 1\ 3]$ is scheduled in slot t, Algo-UD will update the path of the j-th packet to $\mathbf{p}_j = [1\ 3]$. If a packet is delivered, its path is removed from $\mathbf{P}$; see Line 27. Algo-UD terminates when all packets arrive at their destination, i.e., $\mathbf{P} = \emptyset$.

To see how Algo-UD generates a link schedule, consider Figure 1.7. Suppose node 1 2 and 3 have one uplink and one downlink packet. Assume that each node has three antenna elements. Algo-UD starts by initializing $\mathbf{P}$ and F to the empty set and all entries of $\mathbf{a}$ to zero. After recording the path of all packets in $\mathbf{P}$, we then have $\mathbf{P} = <[1\ 4], [4\ 1], [2\ 4], [4\ 2], [3\ 1\ 4], [4\ 1\ 3]>$. Algo-UD then starts to generate the link schedule for slot 1. Algo-UD first sorts nodes in V in decreasing order according to the number of packets buffered at each node. Algo-UD also sorts all packets recorded in $\mathbf{P}$ in decreasing distance order. We then have $\mathbf{v} = [4\ 3\ 2\ 1]$ and $\mathbf{P} =<[4\ 1\ 3], [3\ 1\ 4], [4\ 1], [4\ 2], [2\ 4], [1\ 4]>$. Algo-UD starts with the first node in $\mathbf{v}$. In this case, we find that node 4 has the highest number of buffered packets. All three packets buffered at node 4 is recorded in $\mathbf{p}_1 = [4\ 1\ 3]$, $\mathbf{p}_3 = [4\ 1]$ and $\mathbf{p}_4 = [4\ 2]$, respectively. Algo-UD first checks whether packet p_1 can be transmitted in slot 1.

As scheduling link $e_{4,1}$ in slot 1 satisfies constraints 4, 5 and 6, Algo-UD adds $e_{4,1}$ to F_1. As nodes cannot transmit/receive multiple packets over one link; thus, packet p_3 cannot be transmitted in slot 1. Adding link $e_{4,2}$ in F_1 satisfies constraints 4, 5 and 6. Thus, packet p_4 is transmitted in slot 1. At this point, Algo-UD has checked all three packets buffered at node 4. Algo-UD then moves to node 3, which is the next node in $\mathbf{v}$. There is only one packet, i.e., p_2, buffered at node 3. As activating link $e_{3,1}$ does not violate constraints 4, 5 and 6, Algo-UD adds link $e_{3,1}$ in F_1. Note that activating link $e_{3,1}$ results in NI from node 3 to node 2. As node 2 has a lower order than node 3, node 2 is responsible for canceling the NI caused by node 3. For node 1 and 2, packets p_5 and p_6 cannot be transmitted because node 4 does not have enough antenna elements. After iterating through all nodes in $\mathbf{v}$, Algo-UD adds F_1 into F. Algo-UD then updates the path of each scheduled packets. Note that packet p_4 is delivered in slot 1. Algo-UD then removes $\mathbf{p}_4$ from $\mathbf{P}$. At this point, five packets are yet to be delivered. Algo-UD then starts to generate the schedule for slot 2.

To generate the schedule for slot 2, Algo-UD first sorts all nodes in V where a node that has more buffered packets will have a higher order. Algo-UD also sorts all packets recorded in $\mathbf{P}$ in decreasing distance order. We then have $\mathbf{v}$ = [1 4 2 3] and $\mathbf{P}$ = <[4 1], [2 4], [1 4], [1 4], [1 3] >. Note that the packets p_3 and p_4 have the same path. This is because node 1 has two packets to be delivered to node 4. One of the two packets is node 1's uplink packet, and the other one is the uplink packet from node 3. Algo-UD starts with node 1. Algo-UD adds link $e_{1,4}$ into F_2 to transfer packet p_3. This is because adding link $e_{1,4}$ in F_2 does not violate constraints 4 5 and 6. Similarly, adding link $e_{1,3}$ in F_2 satisfies constraints 4 5 and 6. Thus Algo-UD adds link $e_{1,3}$ to F_2; this link will be used to deliver packet p_5. Packet p_4 cannot be scheduled because each activated link carries one packet. At this point, Algo-UD has checked all three packets buffered at node 1. Algo-UD then moves to node 4, which is the next node in $\mathbf{v}$. Node 4 only has packet p_1 in its buffer. This packet, however, cannot be transmitted because node 1 does not have enough antenna for reception.

Next node in $\mathbf{v}$ is node 2. Algo-UD adds link $e_{2,4}$ in F_2 to deliver packet p_2. The NI between node 2 and 3 is cancelled by node 3; this is because it has a lower order. After scheduling all possible packets in F_2, Algo-UD adds F_2 into F. Algo-UD then updates $\mathbf{P}$ by updating the path of all scheduled packets and removing the path of delivered packets. At this moment, there are packets that have yet to arrive at their destination. Consequently, Algo-UD proceeds to generate the schedule for slot $t = 3$. After sorting node set V and the packets recorded in $\mathbf{P}$, we have $\mathbf{v} = [4\ 1\ 3\ 2]$ and $\mathbf{P} =<[4\ 1], [1\ 4]>$. Links $e_{1,4}$ and $e_{4,1}$ are added to F_3 to deliver packets p_1 and p_2. Algo-UD then adds F_3 into F and removes $\mathbf{p_1}$, $\mathbf{p_2}$ from $\mathbf{P}$. All packets are delivered and Algo-UD terminates. The final schedule F is shown in Table 5.1

Slot 1	**Slot 2**	**Slot 3**
$e_{4,1}$	$e_{1,3}$	$e_{1,4}$
$e_{4,2}$	$e_{1,4}$	$e_{4,1}$
$e_{3,1}$	$e_{2,4}$	

Table 5.1: Schedule for Figure 1.7

Finally, this section gives the computational complexity of Algo-UD. Line $2-4$ has a time complexity of $\mathcal{O}(\sum_{v\in V}(u_v + d_v))$. The complexity of line 9 and 10 is $\mathcal{O}(|V|log(v))$ and $\mathcal{O}(\sum_{v\in V}(u_v+d_v)log\sum_{v\in V}(u_v+d_v))$, respectively. The complexity of line $11-13$ is $\mathcal{O}(|V|)$. In the worst case, only one packet can be scheduled after Algo-UD iterates through all nodes. Thus scheduling all $\sum_{v\in V}(u_v + d_v)$ packets requires a run time complexity of $\mathcal{O}(|V|\cdot\sum_{v\in V}(u_v+d_v))$; see Line $14-25$. To sum up, Algo-UD has a time complexity of $\mathcal{O}(|V|log(v)+\sum_{v\in V}(u_v+d_v)log\sum_{v\in V}(u_v+d_v)+|V|\cdot\sum_{v\in V}(u_v+d_v))$.

5.4 Evaluation

All algorithms are evaluated using Matlab and the Matgraph toolkit [106]. The presented results are an average of 100 simulation runs. For each simulation run, a different topology is used. Also shown in the plots is the confidence interval of 100 simulation runs, where 95% of the results are within the indicated error bar.

The evaluation of UDMAC is presented first before experiment results for Algo-UD are presented in Section 5.4.2. In all experiments, nodes are placed randomly on a 100m $\times$ 100m area, and their transmission range is fixed to 20m. The routing tree is constructed by performing a BFS at the gateway.

5.4.1 UDMAC

The first experiment compares UDMAC against the theoretical upper and lower bound, and also two versions of the state-of-the-art algorithm in [26]. The first version of the algorithm proposed in [26], labeled as ScheTree subsequently, equip nodes with half-duplex radios. As for the second version, labeled as ScheTree+, nodes have full-duplex radios. Each version is run twice; once to forward all uplink packets, and again for downlink packets. Note that in UDMAC, ScheTree and ScheTree+, the transmitter is responsible for canceling NI.

5.4.1.1 Node Density

This set of result concerns different node densities. The number of antenna elements at each node $\Delta_v^{\pm}$ is set to 8. The number of uplink and downlink packets of each node u_v and d_v is randomly chosen from the set {0,1,2,3,4,5} where the number of nodes in the network varies from 30 to 70. From the results shown in Figure 5.2, we see that the difference between UDMAC and the theoretical lower bound is at most 40%. This is because the longest-path rule used by UDMAC may cause inefficient allocation of antenna elements. Nevertheless, UDMAC produces a schedule that is up to 80.6% shorter than ScheTree. In comparison, UDMAC outperforms ScheTree+ by at most 58.6% because ScheTree+ does not exploit the MTR capability of nodes.

5.4.1.2 Node Demand

This set of experiments studies the impact of different node demands; i.e., the number of uplink and downlink packets of each node. The network has 40 nodes

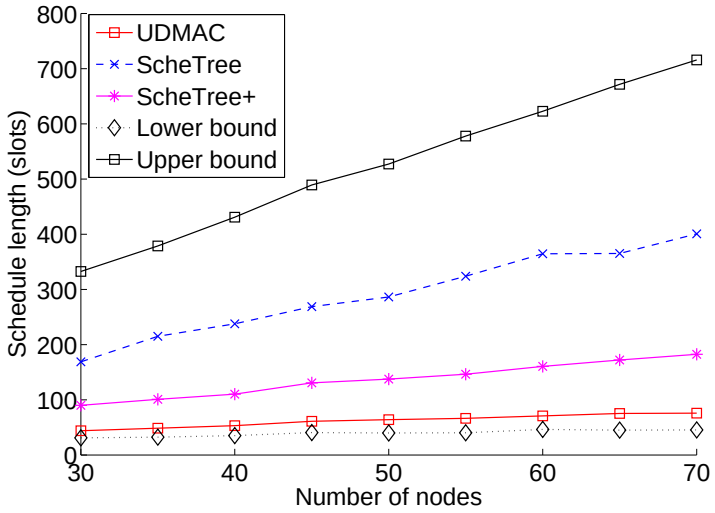


Figure 5.2: Performance evaluation under different node density.

and the number of antenna elements at each node $\Delta_v^{\pm}$ is set to 8. To study varying number of packets at each node, at each point x (number of demands) of Figure 5.3, the value of u_v and d_v is chosen randomly from $[0, x]$. As expected, the schedule length increases proportionally with the number of uplink and downlink packets. The difference between UDMAC and the theoretical lower bound is around 33.8%. As before, UDMAC generates a shorter schedule; up to 77% shorter than ScheTree and 47.9% shorter than ScheTree+.

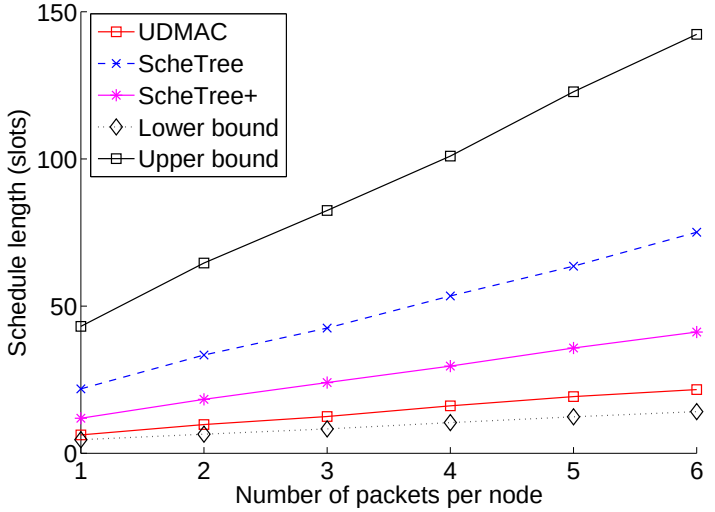


Figure 5.3: Performance evaluation under different node demands.

5.4.1.3 Number of Antennas

Lastly, the performance of UDMAC with different antennas at each node is shown in Figure 5.4. We see that the schedule length decreases with increasing antennas. This is because more antennas equate to more transmitting/receiving streams. When $\Delta^{\pm} \geq 10$, the schedule length no longer decreases because nodes have sufficient antennas to transmit, receive and cancel all interference. The difference between UDMAC and the theoretical lower bound is at most 41.4% when $\Delta^{\pm} = 4$ and reduces to 31% when each node has more than 10 antennas. The schedule length generated by UDMAC is 60% and 17.4% shorter than ScheTree and ScheTree+ respectively when $\Delta^{\pm} = 4$. When each node has $\Delta^{\pm} = 16$ antennas, UDMAC outperforms ScheTree and ScheTree+ by 78.9% and 53.6% respectively.

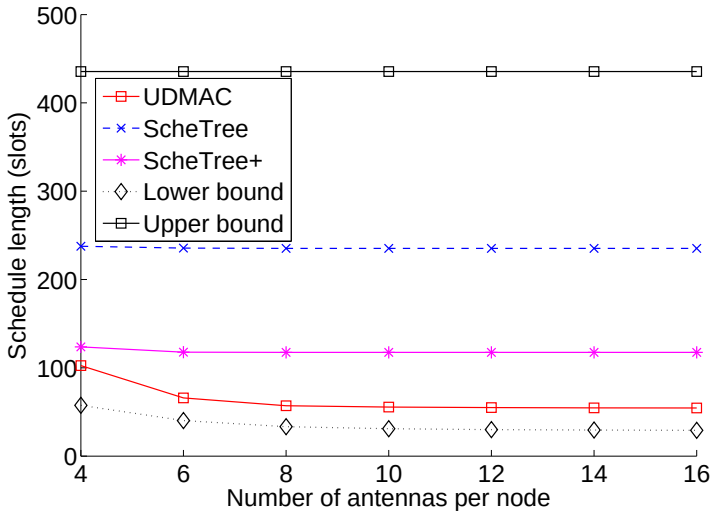


Figure 5.4: Performance evaluation under different number of antennas.

5.4.2 Algo-UD

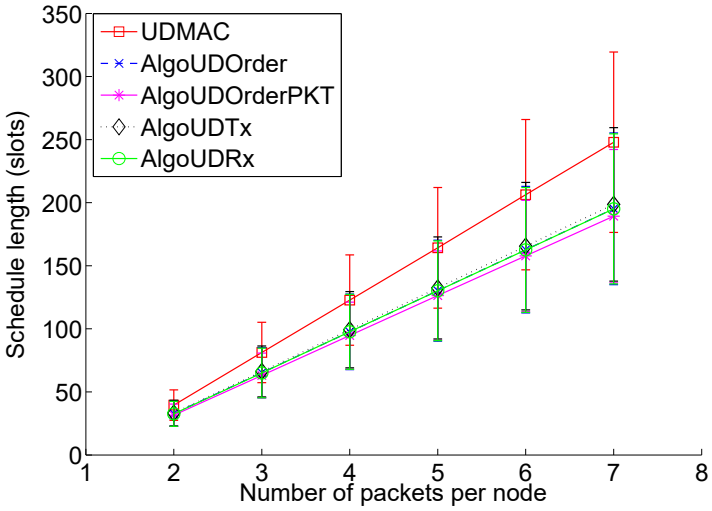
This section compares Algo-UD with UDMAC. In addition, different NI cancellation rules are considered together with Algo-UD; namely Algo-UDTx, Algo-UDRx, Algo-UDOrder and Algo-UDOrderPKT. The difference between them is as follows. Algo-UDTx cancels all NI at the transmitter side only. As for Algo-UDRx, NI is canceled at the receiver side only. Algo-UDOrder and Algo-UDOrderPKT use node ordering

to determine which node cancels NI. Note that different NI cancellation rules are not considered in UDMAC. This is because UDMAC is a path-by-path approach. Multiple time slots are used while scheduling one packet along its shortest path. In this case, the number of DoFs each node uses are distinct in different time slot. Thus a global node order used in different time slot are not appropriate. The order of a nodes is determined by the total number of hops of all its buffered packets. For example, node v has three buffered packets and each packet is two hops away from its destination, thus the total number of hops of all packets buffered at node v is 6. A node with a larger total number of hops has a higher order. However, in Algo-OrderPKT, the order of nodes is only determined by the number of packets buffered at each node. A node with more buffered packets has a higher order.

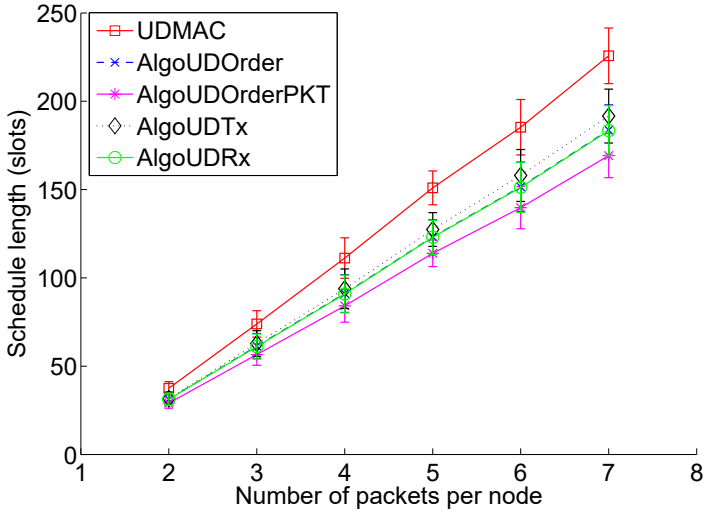
5.4.2.1 Node Demand

This set of results focus on the impact of different node demands; i.e., the number of uplink and downlink packets of each node. The network size is fixed to 40 nodes and the number of antennas at each node is set to 4. The number of packets at each node varies in the two groups of experiments. Specifically, Figure 5.5(a) shows the schedule length where the number of uplink and downlink packets are fixed. On the other hand, in Figure 5.5(b), the value of u_v and d_v is randomly chosen from $[0,\ x]$.

As expected, the schedule length increases proportionally with the number of uplink and downlink packets. Algo-UDOrderPKT has the best performance when the number of uplink/downlink packets at each node is fixed. Algo-UDOrderPKT outperforms UDMAC, Algo-UDTx, Algo-UDRx and Algo-UDOrder by at most 23.6%, 5.1%, 3.4% and 4.1%, respectively. This is because Algo-UD is a greedy algorithm and maximizes the number of packets transmitted in each slot. Further, with node ordering, each node uses its antenna elements more efficiently. This is because a node with more buffered packets has a higher order and NI is cancelled by nodes with a lower order. Consequently, nodes with more buffered packets are able to transmit more packets in each slot. Further, when the number of uplink



(a)



(b)

Figure 5.5: Schedule length versus number of packets per node

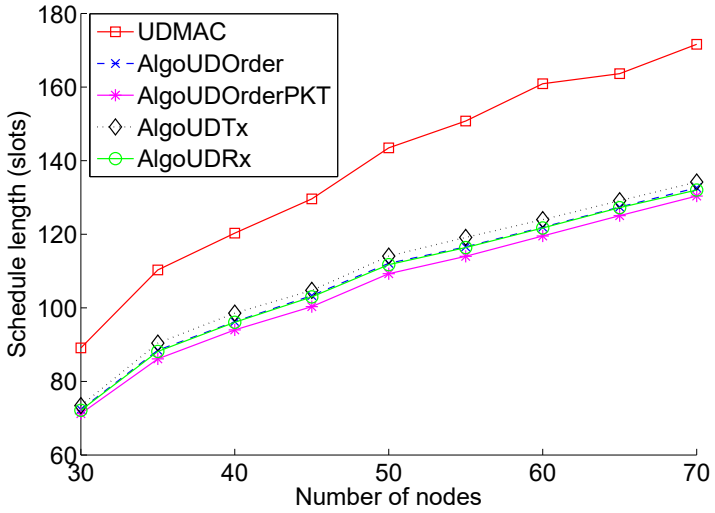
and downlink packets of each node is randomly chosen, Algo-UDOrderPKT still has the best performance. The gap between Algo-UDOrderPKT and UDMAC, Algo-UDTx, Algo-UDRx, Algo-UDOrder is 25.1%, 11.7%, 7.4% and 7.9%, respectively. Note that Algo-UDOrderPKT has a better performance than Algo-UDOrder. This is because in Algo-UDOrderPKT, a node with more buffered packets has a higher order. However, in Algo-UDOrder, a node that has a higher order may not have more buffered packets. Thus, the number of packets scheduled at each slot is not maximized, and the resulting schedule is not the shortest.

5.4.2.2 Node Density

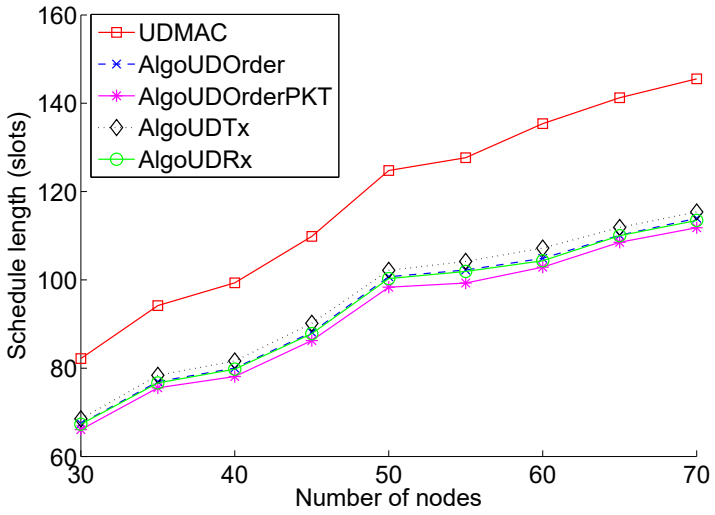
The next experiment concerns the generated schedule length under different node densities. The number of antennas at each node is set to four and the number of nodes in the network varies from 30 to 70. Figure 5.6(a) shows the schedule length under different node densities when each node has 3 uplink packets and 3 downlink packets. Algo-UDOrderPKT has the best performance. The schedule that Algo-UDOrderPKT generates is at most 25.7%, 4.7%, 2.6% and 2.7% shorter than the schedule generated by UDMAC, Algo-UDTx, Algo-UDRx, Algo-UDOrder respectively. When the number of uplink and downlink packets is randomly chosen from zero to five, Algo-UDOrderPKT outperforms UDMAC, Algo-UDTx, Algo-UDRx, Algo-UDOrder by 24%, 4.7%, 2.6% and 2.9%, respectively; see Figure 5.6(b).

5.4.2.3 Number of Antennas

The last group of results show the performance of Algo-UD under different number of antennas per node. The network has 40 nodes and the number of antennas on each node varies from three to eight. Figure 5.7(a) shows the schedule length where each node v has three uplink and three downlink packets. The schedule length decreases as expected when the number of antennas per node decreases. Algo-UDOrderPKT has the best performance. The gap between Algo-UDOrderPKT and UDMAC, Algo-UDTx, Algo-UDRx, Algo-UDOrder is 15.9%, 4.6%, 2.5% and 3.8%, respectively. Figure 5.7(b) shows the schedule length when the number of uplink and downlink packets at each node is randomly chosen between $[0, 5]$. The schedule that Algo-UDOrderPKT generates is at most 20.2%, 5.1%, 3% and 2.2% shorter than the schedule generated by UDMAC, Algo-UDTx, Algo-UDRx, Algo-UDOrder, respectively. When there are eight nodes, all algorithms have almost the same performance. This is because each node has enough antenna elements to transmit, receive and cancel interference.



(a)

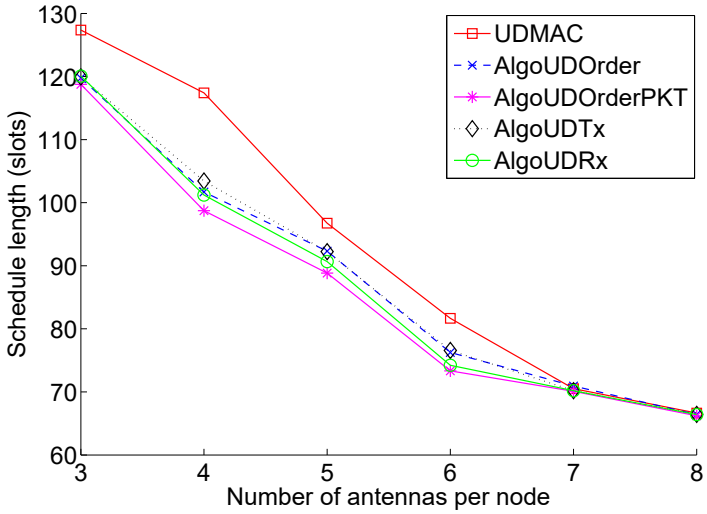


(b)

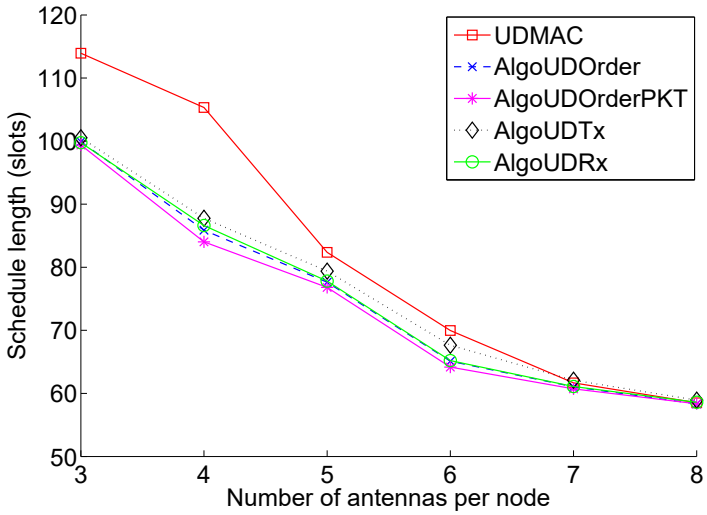
Figure 5.6: Schedule length under different node density

5.5 Conclusion

This chapter investigates scheduling uplink and downlink packets jointly in full-duplex wireless networks. This chapter first presents the theoretical upper and lower bound of the uplink and downlink packet transmission schedule. To address the link scheduling problem, this chapter presents UDMAC and Algo-UD, the first solutions that aims to generate the minimal uplink and downlink packets transmission schedule. Experimental results show that UDMAC generates at most 58.6% shorter UD-Schedule as compared to the state-of-the art half-duplex scheduler with



(a)



(b)

Figure 5.7: Schedule length versus number of antennas per node

full-duplex capability enabled. Moreover, when the number of antenna elements at each node is small, Algo-UD, together with a proper node ordering rule, outperforms UDMAC by at most 25.7%.

Chapter 6

Conclusion

WMNs have many benefits that make them ideal for use as a wireless backbone that interconnects users located in rural, urban and city areas. They, however, are interference limited, which impact their capacity and hence, ability to support the increasing number of users and traffic using the Internet. To this end, this thesis considers a promising approach that equips routers with multiple transmit or/and receive, so called MTR, capability. Advantageously, these routers are able to limit their interference to neighboring nodes and transmit or/and receive *distinct* packets to/from their neighbors. This, however, assumes there is a link scheduler that dictates how and when links are activated. In particular, it determines the links that are active in a given time period and also the frequency in which links are active. Intuitively, the higher number of active links and frequent transmissions translate to high network capacity. Henceforth, this thesis has studied several link scheduling problems in TDMA-based MTR WMNs, and designed several complementary link schedulers that have the overarching goal of improving network capacity. In addition, this thesis has considered both half-duplex and full-duplex nodes.

The first problem addressed is to minimize the superframe or link schedule length in a distributed manner. This problem is significant because a shorter superframe ensures links are activated frequently, and thus ensures high network capacity. In

addition, the derived superframe must meet the transmission requirement of each link. Henceforth, Chapter 3 presents a distributed link scheduler that derives a minimal superframe whilst maximizing the number of links in each slot by solving the NP-complete MAX-CUT problem in every slot. Compared to centralized solutions, Algo-d generates the schedule using only local information. This is advantageous because it improves the scalability and practicality of Algo-d, meaning it is suitable for use in large scale WMNs. Experimental results show that Algo-d achieves similar performance as compared to its centralized counterpart [31] in terms of superframe length and number of concurrent links in each slot. In addition, Algo-d outperforms the following state-of-the-art distributed schedulers: ROMA and JazzyMac.

Data forwarding to/from one or more gateways, with access to the Internet, is a fundamental operation in WMNs. Another significant problem addressed in this thesis is to derive a schedule to forward packets from gateway(s) to their respective destination. Critically, the schedule must be short so that buffered packets are forwarded quickly; i.e., this also means a WMN has high network capacity. To date, past researchers have only considered deriving such a schedule for routers equipped with omni or directional antennas. This thesis, on the other hand, considers routers that incorporate advances in MIMO communications; specifically, interference cancellation and spatial multiplexing. Chapter 4 first considers the single gateway case and proposes a novel centralized link scheduler called Algo-PB that produces a schedule that is within 34.5% of the lower bound. In addition, Algo-PB outperforms state-of-the-art algorithms by at most 45.5%. After that, for the multiple gateways case, aka the forest construction problem, Chapter 4 first presents an Integer Linear Program (ILP) formulation before outlining a heuristic called Algo-FC that generates a balanced forest that is within 9.1% of the ILP solution.

Recent works that address the personalized broadcast or data collection problem only consider half-duplex networks. Thus, the last problem addressed in this thesis is to schedule both uplink and downlink packets jointly in full-duplex wireless networks. This is significant because recent TDMA link schedulers for full-duplex wireless

networks only consider single hop transmissions, and cannot be applied to the joint uplink and downlink packet scheduling problem. To this end, this thesis presents two novel link schedulers, namely UDMAC and Algo-UD, that aim to derive the shortest link schedule that delivers all uplink and downlink packets over multiple hops in full-duplex MU-MIMO based WMNs. Moreover, this thesis considers using a novel node ordering rule to determine which end node of an interference link is required to suppress such interference. This is significant because nodes are likely to have limited DoFs, and an efficient DoFs assignment rule will help reduce schedule length. Experimental results show that UDMAC and Algo-UD outperform state-of-the-art half-duplex link schedulers in terms of schedule length. Moreover, Algo-UD, together with a proper node ordering rule, outperforms UDMAC by at most 25.7%.

In terms of future research, there are many possible directions. For example, Algo-d assumes each link only needs to be activated once. Thus, extending Algo-d to consider weighted links is an interesting future work. This is required in order to adapt to varying link load. Moreover, Algo-d assumes single hop packet transmissions. Thus another possible direction is to design a distributed link scheduler that considers multi-hop packet transmissions in MTR WMNs. Apart from that, Chapter 4 assumes each node has only one path to receive packets from a gateway. This assumption may not be practical as links on a given path may fail. To this end, a future direction is to consider link scheduling and multi-path routing jointly such that a node is able to send/receive packets through another gateway using its backup path when there is a link failure on the primary path. Lastly, a key assumption in Chapter 4 and Chapter 5 is that packet transmissions are reliable; i.e., no consideration for packet loss due to channel errors. Thus another future research direction is to develop a mechanism to handle such packet loss.

Bibliography

[1] P. H. Pathak and R. Dutta, "A Survey of Network Design Problems and Joint Design Approaches in Wireless Mesh Networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 3, pp. 396–428, 2011.

[2] A. D. Gore and A. Karandikar, "Link Scheduling Algorithms for Wireless Mesh Networks," *IEEE Communications Surveys & Tutorials*, vol. 13, no. 2, pp. 258–273, 2011.

[3] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless Mesh Networks: A Survey," *Computer networks*, vol. 47, pp. 445–487, 2005.

[4] J. D. Camp and E. W. Knightly, "The IEEE 802.11s Extended Service Set Mesh Networking Standard," *IEEE Communications Magazine*, vol. 46, no. 8, pp. 120–126, 2008.

[5] S. Yi, Y. Pei, and S. Kalyanaraman, "On the Capacity Improvement of Ad Hoc Wireless Networks Using Directional Antennas," in *ACM international symposium on Mobile ad hoc networking & computing*, (Annapolis, USA), ACM, June 2003.

[6] B. Raman and K. Chebrolu, "Design and Evaluation of a New MAC Protocol for Long-Distance 802.11 Mesh Networks," in *International Conference on Mobile Computing and Networking*, (Cologne, Germany), Aug. 2005.

[7] D. Gesbert, M. Shafi, D.-s. Shiu, P. J. Smith, and A. Naguib, "From Theory to Practice: An Overview of MIMO SpaceCTime Coded Wireless Systems," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 3, pp. 281–302, 2003.

[8] Y. Wang, D. M. Chiu, and J. C. Lui, "Characterizing the Capacity Gain of Stream Control Scheduling in MIMO Wireless Mesh Networks," *Wireless Communications and Mobile Computing*, vol. 9, no. 6, pp. 819–829, 2009.

[9] A. J. Paulraj, D. Gore, R. U. Nabar, H. Bölcskei, *et al.*, "An Overview of MIMO Communications - A Key to Gigabit Wireless," *IEEE Proceedings*, vol. 92, no. 2, pp. 198–218, 2004.

[10] Q. H. Spencer, C. B. Peel, A. L. Swindlehurst, and M. Haardt, "An Introduction to the Multi-User MIMO Downlink," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 60–67, 2004.

[11] E. Perahia, C. Cordeiro, M. Park, and L. L. Yang, "IEEE 802.11 ad: Defining the next generation multi-Gbps Wi-Fi," in *IEEE Consumer Communications and Networking Conference (CCNC)*, (Las Vegas, USA), Jan. 2010.

[12] R. Mudumbai, S. Singh, and U. Madhow, "Medium access control for 60 GHz outdoor mesh networks with highly directional links," in *IEEE INFOCOM*, (Rio de Janeiro, Brazil), Apr. 2009.

[13] Z. Zhang, "Pure Directional Transmission and Reception Algorithms in Wireless Ad Hoc Networks with Directional Antennas," in *IEEE International Conference on Communications (ICC)*, (Seoul, Korea), May 2005.

[14] N. Guo, R. C. Qiu, S. S. Mo, and K. Takahashi, "60-GHzMillimeter-Wave Radio: Principle, Technology, and New Results," *EURASIP Journal on Wireless Communications and Networking*, vol. 2007, no. 1, pp. 48–48, 2007.

[15] S.-H. Wu, L.-K. Chiu, K.-Y. Lin, and T.-H. Chang, "Robust Hybrid Beamforming with Phased Antenna Arrays for Downlink SDMA in Indoor 60 GHz Channels," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4542–4557, 2013.

[16] J. I. Choi, M. Jain, K. Srinivasan, P. Levis, and S. Katti, "Achieving single channel, full duplex wireless communication," in *ACM Mobicom*, (Chicago, USA), Sept. 2010.

[17] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha, "Practical, Real-time, Full-Duplex Wireless," in *ACM Mobicom*, (Las Vegas, USA), Sept. 2011.

[18] X. Xie and X. Zhang, "Semi-synchronous Channel Access for Full-Duplex Wireless Networks," in *IEEE ICNP*, (Raleigh, USA), Oct. 2014.

[19] E. Everett, A. Sahai, and A. Sabharwal, "Passive Self-Interference Suppression for Full-Duplex Infrastructure Nodes," *IEEE Transactions on Wireless Communications*, vol. 13, no. 2, pp. 680–694, 2014.

[20] D. Bharadia, E. McMilin, and S. Katti, "Full Duplex Radios," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 375–386, 2013.

[21] D. Bharadia and S. Katti, "Full Duplex MIMO Radios," in *11th USENIX Conference on Networked Systems Design and Implementation*, (Seattle, USA), Apr. 2014.

[22] D. Kim, H. Lee, and D.-K. Hong, "A Survey of In-band Full-duplex Transmission: From the Perspective of PHY and MAC Layers," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2017 – 2046, 2015.

[23] P. Wang and S. Bohacek, "Practical computation of optimal schedules in multihop wireless networks," *IEEE Transactions on Networking*, vol. 19, pp. 305–315, Apr. 2011.

[24] L. Bao and J. Garcia-Luna-Aceves, "Receiver-Oriented Multiple Access in Ad Hoc Networks with Directional Antennas," *Wireless Networks*, vol. 11, pp. 67–79, 2005.

[25] S. Nedevschi, R. K. Patra, S. Surana, S. Ratnasamy, L. Subramanian, and E. Brewer, "An Adaptive, High performance MAC for Long-Distance Multihop Wireless Networks," in *ACM International Conference on Mobile Computing and Networking*, (San Francisco, USA), Sept. 2008.

[26] J.-C. Bermond, L. Gargano, S. Perénnes, A. A. Rescigno, and U. Vaccaro, "Optimal Time Data Gathering in Wireless Networks with Multi-directional Antennas," *Theoretical Computer Science*, vol. 509, pp. 122–139, 2013.

[27] D. Zhao, K.-W. Chin, and R. Raad, "Approximation Algorithms for Broadcasting in Duty Cycled Wireless Sensor Networks," *Wireless Networks*, vol. 20, no. 8, pp. 2219–2236, 2014.

[28] H. Zeng, Y. Shi, T. Hou, W. Lou, H. Sherali, R. Zhu, and S. Midkiff, "A Scheduling Algorithm for MIMO DoF Allocation in Multi-hop Networks," *IEEE Transactions on Mobile Computing*, vol. 15, no. 2, pp. 264 – 277, 2015.

[29] V. V. Vazirani, *Approximation Algorithms*. Springer, 2004.

[30] K.-W. Chin, S. Soh, and C. Meng, "Novel Scheduling Algorithms for Concurrent Transmit/Receive Wireless Mesh Networks," *Computer Networks*, vol. 56, pp. 1200–1214, 2012.

[31] H.-Y. Loo, S. Soh, and K.-W. Chin, "On Improving Capacity and Delay in Multi Tx/Rx Wireless Mesh Networks," in *IEEE Asia Pacific Conference on Communications (APCC)*, (Bali, Indonesia), Aug 2013.

[32] Y. Shi, J. Liu, C. Jiang, C. Gao, and Y. T. Hou, "An Optimal Link Layer Model for Multi-hop MIMO Networks," in *IEEE INFOCOM*, (Shanghai, China), Apr. 2011.

[33] Y. Ben-David, M. Vallentin, S. Fowler, and E. Brewer, "JaldiMAC: Taking the Distance Further," in *ACM Workshop on Networked Systems for Developing Regions*, (San Francisco, USA), June 2010.

[34] S. K. Hazra and W. K. Seah, "Measurement-Based Link Scheduling for Maritime Mesh Networks with Directional Antennas," *International Journal of Network Management*, vol. 21, pp. 83–105, 2011.

[35] X. Lu, D. Towsley, P. Lio, and Z. Xiong, "An Adaptive Directional MAC Protocol for Ad Hoc Networks using Directional Antennas," *Science China Information Sciences*, vol. 55, no. 6, pp. 1360–1371, 2012.

[36] O. Bazan and M. Jaseemuddin, "A Survey on MAC Protocols for Wireless Ad Hoc Networks with Beamforming Antennas," *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 216–239, 2012.

[37] X. Liu, A. Sheth, M. Kaminsky, K. Papagiannaki, S. Seshan, and P. Steenkiste, "DIRC: Increasing Indoor Wireless Capacity Using Directional Antennas," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 171–182, 2009.

[38] D. Hao and D. P. Liu, "A Distributed TDMA-based MAC Protocol for Ad Hoc Networks with Directional Antennas," in *International Conference on Wireless Communications & Signal Processing (WCSP)*, (Huangshan, China), Oct. 2012.

[39] A. S. Tanenbaum, *Computer Networks*. Prentice Hall, 4-th ed., 2003.

[40] T. Paso, J. Makela, and J. Iinatti, "TDMA Slot Borrowing Scheme Utilizing Smart Antennas in Ad Hoc Networks," in *IEEE Military Communications Conference (MILCOM)*, (San Jose, USA), Oct. 2010.

[41] G. Sanchez, J. Zander, B. Hagerman, *et al.*, "A Reuse-Greedy Algorithm for STDMA Multihop Networks with Advanced Antennas & Rate Control," in *In-*

ternational Symposium on Wireless Pervasive Computing, (Santorini, Greece), May 2008.

[42] M. S. Garache, J. Zander, and B. Hagerman, "Radio Resource Allocation in Spatial TDMA Multihop Networks with Advanced Antennas," in *Wireless Rural and Emergency Communications*, (Rome, Italy), Sept. 2007.

[43] D. Panigrahi and B. Raman, "TDMA Scheduling in Long-Distance WiFi Networks," in *IEEE INFOCOM*, (Rio de Janeiro, Brazil), Apr. 2009.

[44] K. Chebrolu and B. Raman, "FRACTEL: A Fresh Perspective on (Rural) Mesh Networks," in *ACM SIGCOMM Workshop*, (Kyoto, Japan), Aug. 2007.

[45] J. B. Cain, T. Billhartz, L. Foore, E. Althouse, and J. Schlorff, "A Link Scheduling and Ad Hoc Networking Approach using Directional Antennas," in *IEEE Military Communications Conference (MILCOM)*, (Boston, USA), Oct 2003.

[46] Z. Zhang, B. Ryu, G. Nallamothu, and Z. Huang, "Performance of All-Directional Transmission and Reception Algorithms in Wireless Ad Hoc Networks with Directional Antennas," in *IEEE Military Communications Conference (MILCOM)*, (New Jersey, USA), Oct 2005.

[47] G. Jakllari, W. Luo, and S. V. Krishnamurthy, "An Integrated Neighbor Discovery and MAC Protocol for Ad Hoc Networks Using Directional Antennas," *IEEE Transactions on Wireless Communications*, vol. 6, no. 3, pp. 1114–1024, 2007.

[48] X. An and R. Hekmat, "Directional MAC Protocol for Millimeter Wave based Wireless Personal Area Networks," in *Vehicular Technology Conference, VTC Spring*, (Singapore), May 2008.

[49] I. K. Son, S. Mao, M. X. Gong, and Y. Li, "On Frame-based Scheduling for

Directional mmWave WPANs," in *IEEE INFOCOM*, (Orlando, USA), Mar. 2012.

[50] J. Qiao, L. X. Cai, J. W. Mark, *et al.*, "STDMA-Based Scheduling Algorithm for Concurrent Transmissions in Directional Millimeter Wave Networks," in *IEEE International Conference on Communications (ICC)*, (Ottawa, Canada), June 2012.

[51] J. Qiao, L. X. Cai, X. S. Shen, and J. W. Mark, "Enabling Multi-Hop Concurrent Transmissions in 60 GHz Wireless Personal Area Networks," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 3824–3833, 2011.

[52] A. Capone, I. Filippini, and F. Martignon, "Joint Routing and Scheduling Optimization in Wireless Mesh Networks with Directional Antennas," in *IEEE International Conference on Communications*, (Beijing, China), May 2008.

[53] H. Lu, S. Liu, and A. Jiang, "A Cross-Layer Design for End-to-End On-Demand Bandwidth Allocation in Infrastructure Wireless Mesh Network," in *International Conference on Wireless Algorithms, Systems and Applications*, (Chicago, USA), Aug. 2007.

[54] V. Ramamurthi, A. Reaz, S. Dixit, and B. Mukherjee, "Link Sheduling and Power Control in Wireless Mesh Networks with Directional Antennas," in *IEEE International Conference on Communications (ICC)*, (Beijing, China), May 2008.

[55] A. Spyropoulos and C. S. Raghavendra, "Energy Efficient Communications in Ad Hoc Networks Using Directional Antennas," in *IEEE INFOCOM*, (New York, USA), June 2002.

[56] K.-W. Chin, S. Soh, and C. Meng, "A Novel Scheduler for Concurrent Tx/Rx Wireless Mesh Networks with Weighted Links," *IEEE Communications Letters*, vol. 16, pp. 246–248, 2012.

[57] H.-N. Dai, S. C. Liew, and L. Fu, "Link Scheduling in Multi-Transmit-Receive Wireless Networks," in *IEEE Conference on Local Computer Networks (LCN)*, (Bonn, Germany), Oct. 2011.

[58] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad-Hoc Networks," in *ACM nternational ymposium on Mobile ad hoc Networking and Computing*, (Florence, Italy), May 2006.

[59] A. Das and T. Zhu, "A Reservation-Based TDMA MAC Protocol using Directional Antennas (RTDMA-DA) for Wireless Mesh Networks," in *IEEE Global Telecommunications Conference (GLOBECOM)*, (Washington, DC, USA), Nov. 2007.

[60] J. Crichigno, M.-Y. Wu, S. K. Jayaweera, and W. Shu, "Throughput Optimization in Multihop Wireless Networks with Multi-packet Reception and Directional Antennas," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1206–1213, 2011.

[61] P. Dutta, V. Mhatre, D. Panigrahi, and R. Rastogi, "Joint Routing and Scheduling in Multi-hop Wireless Networks with Directional Antennas," in *IEEE INFOCOM*, (San Diego, USA), Mar. 2010.

[62] L. Wang, K.-W. Chin, S. Soh, and R. Raad, "Novel Joint Routing and Scheduling Algorithms for Minimizing End-to-End Delays in Multi Tx-Rx Wireless Mesh Networks," *Computer Communications*, vol. 72, pp. 63 – 77, 2015.

[63] P. Dutta, S. Jaiswal, D. Panigrahi, and R. Rastogi, "A New Channel Assignment Mechanism for Rural Wireless Mesh Networks," in *IEEE INFOCOM*, (Phoenix, USA), Apr. 2008.

[64] H. Su and X. Zhang, "Joint Link Scheduling and Routing for Directional-

Antenna Based 60 GHz Wireless Mesh Networks," in *IEEE Global Telecommunications Conference, GLOBECOM*, (Hawaii, USA), Sept. 2009.

[65] G. K. Audhya, K. Sinha, S. C. Ghosh, and B. P. Sinha, "A Survey on the Channel Assignment Problem in Wireless Networks," *Wireless Communications and Mobile Computing*, vol. 11, no. 5, pp. 583–609, 2011.

[66] H. Skalli, S. Ghosh, S. K. Das, L. Lenzini, and M. Conti, "Channel Assignment Strategies for Multiradio Wireless Mesh Networks: Issues and Solutions," *IEEE Communications Magazine*, vol. 45, no. 11, pp. 86–95, 2007.

[67] M. Yazdanpanah, C. Assi, and Y. Shayan, "Optimal Joint Routing and Scheduling in Wireless Mesh Networks with Smart Antennas," in *IEEE International Symposium on World of Wireless Mobile and Multimedia Networks (WoWMoM)*, (Montreal, Canada), June 2010.

[68] M. E. Steenstrup, "Exploiting Directional Antennas to Provide Quality of Service and Multipoint Delivery in Mobile Wireless Networks," in *IEEE Military Communications Conference (MILCOM)*, (Boston, USA), Oct 2003.

[69] K. Sundaresan, W. Wang, and S. Eidenbenz, "Algorithmic Aspects of Communication in Ad-hoc Networks with Smart Antennas," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, (Florence, Italy), May 2006.

[70] S. Chu and X. Wang, "Opportunistic and Cooperative Spatial Multiplexing in MIMO Ad Hoc Networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 18, no. 5, pp. 1610–1623, 2010.

[71] C.-T. Chang, C.-Y. Chang, and Y.-J. Lu, "Maximizing Throughput by Exploiting Spatial Reuse Opportunities with Smart Antenna Systems," in *IEEE International Conference on Communications (ICC)*, (Cape Town, South Africa), May 2010.

[72] L.-L. Hung, S.-H. Wu, and C.-T. Chang, "Maximizing Throughput for Delay-Constraint Transmissions with Smart Antenna Systems in WLANs," in *International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, (Seoul, Korea), June 30-July 02 2011.

[73] I. Jawhar, J. Wu, and D. P. Agrawal, "Resource Scheduling in Wireless Networks using Directional Antennas," *IEEE Transactions on Parallel and Distributed Systems*, vol. 21, pp. 1240–1253, 2010.

[74] B. Mumey, J. Tang, and T. Hahn, "Joint Stream Control and Scheduling in Multihop Wireless Networks with MIMO Links," in *IEEE International Conference on Communications*, (Beijing, China), May 2008.

[75] A. Deopura and A. Ganz, "Provisioning Link Layer Proportional Service Differentiation in Wireless Networks with Smart Antennas," *Wireless Networks*, vol. 13, no. 3, pp. 371–378, 2007.

[76] R. Bhatia and L. E. Li, "Throughput Optimization of Wireless Mesh Networks with MIMO Links," in *IEEE INFOCOM*, (Anchorage, USA), May 2007.

[77] X. Wang and J. Garcia-Luna-Aceves, "Embracing Interference in Ad Hoc Networks using Joint Routing and Scheduling with Multiple Packet Reception," *Ad Hoc Networks*, vol. 7, no. 2, pp. 460–471, 2009.

[78] P. Huang, L. Xiao, S. Soltani, M. W. Mutka, and N. Xi, "The evolution of mac protocols in wireless sensor networks: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 101–120, 2013.

[79] V. Bonifaci, P. Korteweg, A. Marchetti-Spaccamela, and L. Stougie, "An Approximation Algorithm for the Wireless Gathering Problem," *Operations Research Letters*, vol. 36, no. 5, pp. 605–608, 2008.

[80] H. Zhang, P. Soldati, and M. Johansson, "Performance Bounds and Latency-

Optimal Scheduling for Convergecast in WirelessHART Networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 6, pp. 2688–2696, 2013.

[81] Z. Shen, H. Jiang, and Z. Yan, "Fast Data Collection in Linear Duty-Cycled Wireless Sensor Networks," *IEEE Transactions on Vehicular Technology*, vol. 63, no. 4, pp. 1951–1957, 2014.

[82] H. Choi, J. Wang, and E. A. Hughes, "Scheduling for Information Gathering on Sensor Network," *Wireless Networks*, vol. 15, no. 1, pp. 127–140, 2009.

[83] J. Zhao, Y. Qin, D. Yang, and Y. Rao, "A Source Aware Scheduling Algorithm for Time-Optimal Convergecast," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014.

[84] O. D. Incel, A. Ghosh, B. Krishnamachari, and K. Chintalapudi, "Fast Data Collection in Tree-Based Wireless Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 11, no. 1, pp. 86–99, 2012.

[85] B. Malhotra, I. Nikolaidis, and M. A. Nascimento, "Aggregation Convergecast Scheduling in Wireless Sensor Networks," *Wireless Networks*, vol. 17, no. 2, pp. 319–335, 2011.

[86] Y.-F. Wen and F. Y.-S. Lin, "The Top Load Balanced Forest Routing in Mesh Networks," in *IEEE Consumer Communications and Networking Conference*, (Las Vegas), Jan. 2006.

[87] S. Das, K. Papagiannaki, S. Banerjee, and Y. Tay, "SWARM: The Power of Structure in Community Wireless Mesh Networks," *IEEE/ACM Transactions on Networking (TON)*, vol. 19, no. 3, pp. 760–773, 2011.

[88] X. Xu, W. Liang, X. Jia, and W. Xu, "Network Throughput Maximization in Unreliable Wireless Sensor Networks with Minimal Remote Data Transfer Cost," *Wireless Communications and Mobile Computing*, p. Accepted, 2015.

[89] V. Mhatre, H. Lundgren, F. Baccelli, and C. Diot, "Joint MAC-aware routing and load balancing in mesh networks," in *ACM CoNEXT*, (New York), Dec. 2007.

[90] D. Nandiraju, L. Santhanam, N. Nandiraju, and D. P. Agrawal, "Achieving Load Balancing in Wireless Mesh Networks Through Multiple Gateways," in *IEEE International Conference on Mobile Adhoc and Sensor Systems (MASS)*, (Vancouver), Oct 2006.

[91] S. Maurina, R. Riggio, T. Rasheed, and F. Granelli, "On tree-based routing in multi-gateway association based wireless mesh networks," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications*, (Tokyo), Sept. 2009.

[92] Y. Yan, L. Ci, R. Zhang, and Z. Wang, "Load Balancing Routing Algorithm among Multiple Gateways in MANET With Internet Connectivity," in *IEEE International Conference on Advanced Communication Technology (ICACT)*, (Pyeongchang, Korea), Feb. 2014.

[93] K. Miura and M. Bandai, "Node Architecture and MAC Protocol for Full Duplex Wireless and Directional Antennas," in *IEEE International Symposium on Personal Indoor and Mobile Radio Communications (PIMRC)*, (Sydney, Australia), Sept. 2012.

[94] K. Tamaki, H. Ari Raptino, Y. Sugiyama, M. Bandai, S. Saruwatari, and T. Watanabe, "Full Duplex Media Access Control for Wireless Multi-hop Networks," in *IEEE Vehicular Technology Conference*, (Dresden, Germany), June 2013.

[95] Y. Sugiyama, K. Tamaki, S. Saruwatari, and T. Watanabe, "A Wireless Full-duplex and Multi-hop Network with Collision Avoidance using Directional Antennas," in *International Conference on Mobile Computing and Ubiquitous Networking (ICMU)*, (Singapore), Jan. 2014.

[96] N. Singh, D. Gunawardena, A. Proutiere, B. Radunovic, H. V. Balan, and P. Key, "Efficient and Fair MAC for Wireless Networks with Self-interference Cancellation," in *International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt)*, (Princeton, USA), May 2011.

[97] E. Askari and S. Aïssa, "Single-Band Full-Duplex MAC Protocol for Distributed Access Networks," *IET Communications*, vol. 8, no. 10, pp. 1663–1673, 2014.

[98] J. Y. Kim, O. Mashayekhi, H. Qu, M. Kaz, and P. Levis, "Janus: A Novel MAC Protocol for Full Duplex Radio." Stanford University, Technical Report, 2013.

[99] S. Queiroz and R. Hexsel, "Translating Full Duplexity into Capacity Gains for the High-Priority Traffic Classes of IEEE 802.11," in *ACM Symposium on Applied Computing*, (Salamanca, Spain), Apr. 2015.

[100] A. Tang and X. Wang, "A-Duplex: Medium Access Control for Efficient Coexistence Between Full-Duplex and Half-Duplex Communications," *IEEE Transactions on Wireless Communications*, vol. 14, no. 10, pp. 5871–5885, 2015.

[101] E. Aryafar and A. Keshavarz-Haddad, "FD 2: A Directional Full Duplex Communication System for Indoor Wireless Networks," in *IEEE Conference on Computer Communications (INFOCOM)*, pp. 1993–2001, IEEE, 2015.

[102] J. Lee, W. Kim, S.-J. Lee, D. Jo, J. Ryu, T. Kwon, and Y. Choi, "An Experimental Study on the Capture Effect in 802.11a Networks," in *ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, (Montreal, Canada), Sept. 2007.

[103] A. Kochut, A. Vasan, A. U. Shankar, and A. Agrawala, "Sniffing Out the Correct Physical Layer Capture Model in 802.11b," in *International Conference on Network Protocols*, (Berlin, Germany), Oct. 2004.

[104] M. Maróti, B. Kusy, G. Simon, and Á. Lédeczi, "The Flooding Time Synchronization Protocol," in *ACM International Conference on Embedded Networked Sensor Systems*, (Baltimore, USA), Nov. 2004.

[105] M. X. Goemans and D. P. Williamson, "Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming," *Journal of the ACM*, vol. 42, no. 6, pp. 1115–1145, 1995.

[106] E. R. Scheinerman, "Matgraph: a MATLAB Toolbox for Graph Theory," *Software available at: http://www. ams. jhu. edu/˜ ers/matgraph*, 2007.

[107] L. Gargano and A. A. Rescigno, "Optimally Fast Data Gathering in Sensor Networks," in *Mathematical Foundations of Computer Science*, pp. 399–411, Springer, 2006.

[108] P. W. Wolniansky, G. J. Foschini, G. Golden, R. Valenzuela, *et al.*, "V-BLAST: An Architecture for Realizing Very High Data Rates over the Rich-scattering Wireless Channel," in *URSI International Symposium on Signals, Systems, and Electronics*, (Pisa, Italy), Oct. 1998.