2015

Link scheduling in multi-transmit-receive wireless mesh networks

Luyao Wang
University of Wollongong, lw233@uowmail.edu.au

Link Scheduling in Multi-Transmit-Receive Wireless Mesh Networks

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

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Luyao Wang

Bachelor of Engineering (Telecommunications)

School of Electrical, Computer and Telecommunications Engineering

July 2015

Statement of Originality

I, Luyao 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

Luyao Wang

July, 2015

Abstract

A Wireless Mesh Network (WMN) is comprised of mesh routers and mesh clients that are able to self-organize into an arbitrary network topology. Routers in WMNs are able to send/receive packets to/from the Internet, and relay packets for other nodes. Hence, WMNs are widely used to improve network coverage, and are ideal as a communication backbone that serves users in metropolitan as well as rural areas. However, their capacity is of concern to operators and researchers. A promising approach to increase their capacity is to equip each router with multiple transmit (Tx) or receive (Rx) capability, aka MTR. This can be achieved for example using multiple off-the-shelf IEEE 802.11 radios and parabolic antennas, 60 GHz radios or Multi-User Multiple Input Multiple Output (MU-MIMO) wireless technologies. In these systems, nodes operate over a *single* frequency and can transmit to or receive from multiple distinct neighbors concurrently. However, they will experience collision when they transmit *and* receive at the same time.

This thesis thus aims to develop novel link schedulers that exploit the MTR capability of nodes to yield a high capacity WMN. Specifically, it targets spatial Time Division Multiple Access (TDMA) based link schedulers that are designed to meet one or more objectives. Briefly, a TDMA schedule or superframe consists of a number of time slots. In each slot, a set of non-interfering links are active. As compared with random access methods, a TDMA schedule provides collision-free

and deterministic channel access. Moreover, if the resulting superframe generated by a link scheduler is short, meaning links are activated frequently, the network capacity will be high.

To date, existing MTR schedulers only aim to maximize network capacity without considering the Quality of Service (QoS) of end-to-end flows. This is important for real-time multimedia applications. Hence, this thesis considers the following problem: given a set of end-to-end traffic demands in an MTR WMN, derive a schedule that minimizes the total delay of these demands. Deriving such a schedule, however, is challenging because the end-to-end delay of a flow is affected by routing, superframe length and transmission order or links. Thus, this thesis proposes two heuristic algorithms: JRS-Multi-DEC and JRS-BIP. Both algorithms find the optimal routing that minimizes the maximum link load and also yields a minimal superframe length. A novel slot re-ordering algorithm is then used to further reduce end-to-end delays.

Fairness is also an important objective. A scheduler that aims to maximize throughput without considering fairness may starve flows. However, no existing MTR schedulers have considered fairness when scheduling links. Thus, a challenging link scheduling problem is how to generate a superframe that ensures end-to-end fairness while also maximizes network capacity. Hence, this thesis outlines a novel link scheduler called Algo-Fair. It uses a novel augmentation step to distribute spare capacity fairly amongst flows. It is the first link scheduler for MTR WMNs that jointly maximizes network capacity and the fairness of flow rates. Critically, unlike past works, it does not require a conflict graph or a Linear Program (LP) to first calculate a fair share before deriving a schedule, which may not exist. Algo-Fair, on the other hand, generates a superframe directly and ensures flow rates approximate the well-known max-min fair criterion.

A key assumption in existing MTR link schedulers is that traffic information is fixed and known in advance. However, in practice, traffic information is likely to be random. Thus, a pre-computed superframe may lead to excessive idle times

or collisions. In this respect, when the traffic load is low, random access is ideal because a link can relinquish the channel when it has finished transmission. On the other hand, for high traffic load, a TDMA schedule is preferred because it provides deterministic and collision-free channel access. Hence, this thesis considers dimensioning a hybrid superframe that contains both a TDMA and random access part. A key challenge when sizing these two parts is to minimize idle time and collision. To address this problem, this thesis presents the first Stochastic Programming (SP) based approach. In particular, the SP approach has a penalty parameter that can be used in conjunction with binary search to construct a hybrid superframe that minimizes idle time whilst satisfying a desired probability of collision.

Another problem with regards to random demands is that a derived superframe may need to be updated frequently if demands change. Critically, this will require a central controller to collect queue information, compute a new schedule, and disseminated it to all nodes frequently. Unfortunately, this will cause high signaling overheads, especially in large scale multi-hop WMNs. Therefore, a challenging problem is how to route flows and construct a minimum superframe that is robust against random demands. Thus, this thesis contains a novel heuristic algorithm called Algo-PolyH. It uses four LPs to iteratively generate a robust routing and superframe solution that supports all demands characterized by a polyhedral set. This is confirmed by experiments over networks with different number of flows, nodes, available paths and node degrees.

Lastly, this thesis considers MU-MIMO-based MTR WMNs, whereby some antennas are used for data streams whilst others cancel interference to/from neighboring links. Each interference stream must be nulled/suppressed at the transmitter and/or receiver sides. Hence, a fundamental problem is deriving a rule that efficiently balances the number of antennas that are used for data and interference cancellation. This thesis shows that a recently developed node ordering rule produces the shortest possible superframe amongst all competing rules.

Acknowledgments

First and foremost, I would like to thank my supervisor, Associate Professor Kwan-Wu Chin, for his patience, selfless guidance, constructive suggestions and constant encouragement. He has taught me how to find and solve a research problem as well as how to produce a high quality paper. Without his contribution in terms of time, ideas and hard work, it would be impossible for me to finish my PhD degree or get any publications. I appreciate his strict and high requirements that have motivated me to grow into an independent researcher. On the other hand, his kindness and sense of humor have helped me overcome difficulties and frustrations experienced throughout my PhD studies. I am also thankful for the excellent example he has set as a good researcher and as a responsible supervisor. His advice about research and life are precious treasure for my future.

Special thanks to my co-supervisors, Dr Raad Raad and Dr Sieteng Soh, for their contributions and helpful comments in the past three years.

I also want to thank all the anonymous reviewers and editors of conferences and journals for their time, effort and comments. Their suggestions have certainly raised the quality of my publications.

My PhD studies at University of Wollongong was made enjoyable in large part because of the many friends and my dearest roommates. Special thanks go to my friends and research mates in Building 39A.201, Mr He Wang, Mr Changlin Yang,

Mr Shichao Fu, Mr Sen Zhang and Mr Tengjiao He, for countless joyful conversations and pleasant working environment. Thanks to my roommates, Miss Jiaying Lu and Miss Manruoshan Wu, for their understanding and delicious meals when I was busy with my research.

Lastly, I would like to thank my parents for all their love and encouragement. They always believe in my abilities and are eternally proud of me; all these have motivated me to be a better person throughout my life. Without their support and understanding, I could not achieve this stage.

Contents

List of Figures

List of Tables

Abbreviations

BDA	Bucket Draining Algorithm
BIP	Binary Integer Programming
CSI	Channel State Information
DEC	Directed Edge Coloring
DoF	Degree of Freedom
IAT	Inter-link Activation Time
IC	Interference Cancellation
ILP	Integer Linear Programming
JRS	Joint Routing and Scheduling
LP	Linear Programming
MAC	Medium Access Control
MCs	Mesh Clients
MILP	Mixed Integer Linear Program
MIMO	Multiple Input Multiple Output
MINLP	Mixed Integer Non-Linear Program
MIP	Mixed-integer Programming
MMF	Max-min Fair
MMSE-SIC	Minimum Mean Square Error SIC
MPR	Multiple Packet Reception

MRs	Mesh Routers
MTR	Multi-Transmit-Receive
NILP	Non-linear Integer Programming
NLP	Non-linear Program
QoS	Quality of Service
SAA	Sample Average Approximation
SINR	Signal to Interference and Noise Ratio
SDMA	Spatial Division Multiple Access
SIC	Successive Interference Cancellation
SP	Stochastic Programming
TDMA	Time Division Multiple Access
TMs	Traffic Matrices
WCD	Worst-case Delay
WMN	Wireless Mesh Network

Chapter 1

Introduction

1.1 Background

Wireless Mesh Networks (WMNs) are a form of ad hoc networks where nodes dynamically self-organize into an arbitrary network topology [1]. Advantageously, they can be used to improve the coverage, capacity and reliability of an existing infrastructure [2]. Figure 1.1 shows a hybrid WMN, comprising of Mesh Routers (MRs) and Mesh Clients (MCs). The MRs form a communication backbone; two of which have connectivity to the Internet whilst others may be equipped with multiple interfaces that enable them to function as gateways for other wireless networks; e.g., WiMAX [3]. MRs are responsible for forwarding packets to/from the Internet and other nodes that are part of a WMN. This means MRs carry most of the traffic. MCs can be laptops, an iPhone™ or an iPad™, and in some cases function as a router. In a nutshell, WMNs have the following characteristics:

- **Multi-hop**. Nodes rely on each other to forward packets. Nodes can be deployed on an as-needed basis and use multi-hop communications to gain access to the Internet or other wireless networks [4].

- **Self-organization and self-configuration**. Nodes do not rely on a central entity for routing or to discover each other. New MRs can automatically

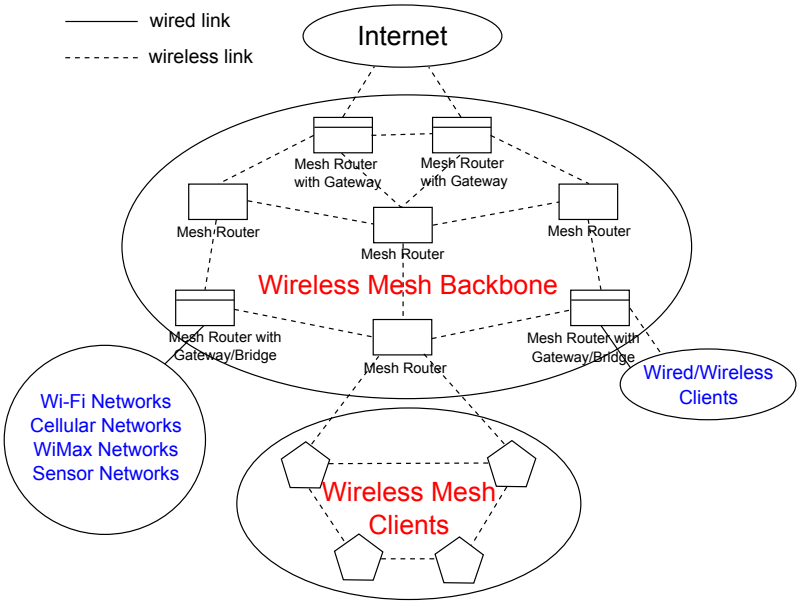


Figure 1.1: A typical wireless mesh network

connect to existing MRs quickly and start serving MCs. Moreover, they are able to adapt to topological changes; e.g., make use of multiple paths as new MRs are installed.

- **Heterogeneous network access**. From Figure 1.1, we see that WMNs can also include connectivity to Wireless Local Area Networks (WLANs) [5], WiMAX [3] and other wireless networks. They support both backhaul access to the Internet and Peer-to-Peer (P2P) communications [6]. In addition, MRs can be equipped with multiple radios to improve network capacity or to bridge different wireless technologies.

- **Mobility**. As mentioned earlier, MCs can be stationary or mobile. WMNs can therefore support users in vehicles such as cars, trains and ferries [7]. These nodes, however, may have limited resources; e.g., finite battery capacity.

Example WMNs include Technology For All (TFA) Rice Mesh [8], Wi-Fi Based Long Distance Networks (WiLDNet) [9], MIT Roofnet [10], LokVaani [11], Feng Chia University (FCU) Wireless Network [12] and Aruba Mesh [13]. Table 1.1 summarizes these WMNs. The IEEE has also standardized two technologies to support meshing. The first is IEEE 802.11s [14]. It allows wireless nodes to inter-

Examples	Number of MRs	Wireless Link	Antennas	Applications
TFA Rice Mesh [8]	18	IEEE 802.11b	Omni-directional antennas	High-performance ultra-low cost wireless access; low-cost health sensing and communication applications
WiLDNet [9]	4	IEEE 802.11a/b/g	High-gain directional antennas [16]	Voice service to rural areas in India
MIT Roofnet [10]	50	IEEE 802.11b/g	Three directional Yagi antennas and omni-directional antennas	Provide Internet access to students
LokVaani [11]	7	IEEE 802.15.4	Tmote-Sky Platforms with an MSP 430 microcontroller and CC2420 radio	Enable local voice communication in developing regions or on-site, local communication among users in emergency situations
FCU Wireless Network [12]	400	IEEE 802.16 and 802.11b	Antennas with 60-degree beamwidth	Improve access/coverage and quality of wireless networks in FCU
Aruba Mesh [13]	1000	IEEE 802.11n	Omni-directional antennas	Coverage of areas where wired connectivity is impractical or unavailable

Table 1.1: A comparison of commercial and experimental WMNs

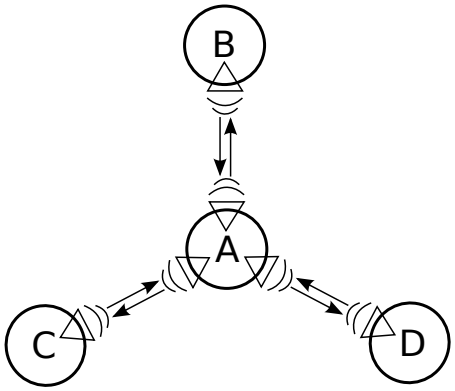
connect and form a multi-hop wireless network without the help of an Access Point (AP) [14]. The second is IEEE 802.16, aka WiMAX [3]. This standard supports two transmission modes: Point-to-Multipoint (PMP) and mesh [3]. In PMP, traffic is transmitted between a Base Station (BS) and Subscriber Stations (SSs). However, in mesh mode, traffic can also be relayed between SSs [15].

The capacity of WMNs is of key concern to network operators and researchers. In [17], the capacity of multi-hop wireless networks is shown to scale according to $\Theta(\frac{W}{\sqrt{nlogn}})$ under the protocol and physical model [18], where n is number of

nodes and W is the number of bits that each node can transmit in one second. The upper bound on throughput is $\Theta(\frac{W}{\sqrt{n}})$ for random networks and $\Theta(\frac{W}{n^{\frac{1}{\alpha}}})$ for arbitrary networks, where α is the decay rate of the transmitted signal; note, nodes are equipped with an omni-directional antenna. The results can be explained as follows. First, nodes need to share the channel with neighbouring nodes, meaning the capacity decreases with increasing number of nodes. Second, increasing their load means other nodes forward more traffic, which further decreases network capacity. Thirdly, the transmission range of each node has a significant impact on the number of nodes sharing a channel.

The poor scalability and capacity of WMNs, as implied by the theoretical results of [17], have spurred extensive research into new approaches that encompas physical layer techniques, link scheduling or random access methods and routing to name a few. This thesis contributes to this global effort. It considers approaches that endow routers with multi transmit (Tx) or receive (Rx) capability. Such routers have N antennas/radios and they are able to initiate up to N transmissions/receptions to/from N distinct neighbors concurrently. In particular, they have the ability to null/suppress/ignore interference but cannot transmit and receive at the same time; this is referred to as the no *mix-tx-rx* constraint. Advantageously, a node will not experience collision when two or more neighbors transmit to it. Referring to Figure 1.2, we see that node A is able to transmit/receive to/from three neighbors in time slot 1 and 2, respectively.

To date, there are three systems with MTR capability. Firstly, nodes can be equipped with multiple radios and high-gain directional antennas operating on the same frequency. Briefly, directional antennas increase network capacity because as compared to omni-directional antennas, they have a longer range, smaller interference footprint, and higher spatial reuse [19] [20]. To this end, in [21], the authors equip nodes with multiple off-the-shelf IEEE 802.11 radios, each connected to a parabolic antenna. The resulting system is then used to provide long-distance, tens of kilometres, point-to-point links. In order to achieve MTR capability, the authors



Slot 1	Slot 2
A→B	B→A
A→C	C→A
A→D	D→A

Figure 1.2: An example MTR wireless network and its Time Division Multiple Access (TDMA) schedule

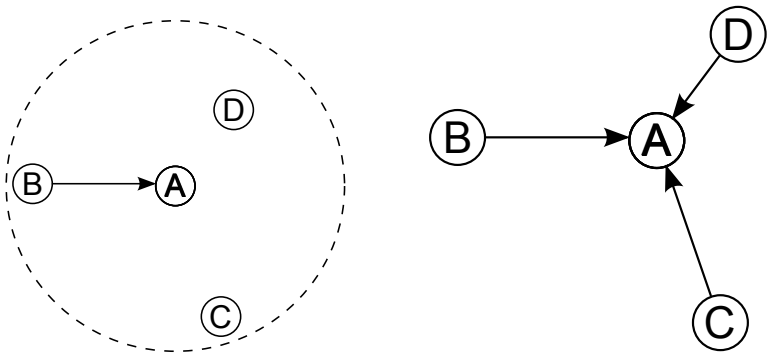
use transmit power control, ensuring incident links are separated by at least 30°, and removing carrier sense so that a node can initiate multiple concurrent transmissions. Note that the 30° link angle constraint can be relaxed using polarized antennas. In addition, nodes use transmission power control to ensure each link has the minimum signal to interference and noise ratio (SINR). Note, the transmission power is fixed at network setup time; i.e., each node sets its transmit power to a value that enables concurrent receptions.

Secondly, nodes can be equipped with 60 GHz radios. The key feature of the 60 GHz band is its high directivity. Moreover, the use of flat-top antennas means the interference between neighboring links can be ignored. In fact, the authors of [22] concluded that mm-wave (60 GHz) wireless links can be considered as *pseudo-wires*. Critically, the authors show that if the links are highly directional, the interference caused by neighboring transmissions can be ignored in both the physical and protocol interference model [18].

Lastly, an MTR WMN can be realized using Multiple Input Multiple Output (MIMO) technology. Specifically, each node has an antenna array, where the signal received and transmitted by each antenna element is intelligently combined or separated to yield higher gains or to remove co-channel interference. MIMO can increase signal range, suppress interference, and combat signal fading; all of which increase network capacity [23]. Apart from that, MIMO is best suited for multi-

path environments because it provides extremely high spectral efficiencies. Notably, MIMO technology allows a node to simultaneously transmit/receive multiple independent data streams over the same frequency band [24]. In [25], a sender is able to concurrently transmit independent data streams to multiple neighbors over different antenna elements. Receiver nodes can then use minimum mean square error sequential interference cancellation (MMSE-SIC) to decode multiple streams. A key assumption is that channel state information (CSI) is available at transmitting and receiving nodes. This is reasonable given that nodes in MTR WMNs are primarily static and pilot symbols can be transmitted periodically to learn the CSI. Nodes are typically assumed to have equal transmission power that is divided equally amongst their antenna elements. Apart from that, a node can use a subset of its antenna elements or Degree of Freedoms (DoFs) to null/suppress interference from neighboring transmissions.

As noted earlier, a key characteristic of MTR systems is that nodes are able to transmit or receive from neighbors simultaneously. This in turn creates a fundamentally different interference model as compared to past works that assume the physical or protocol interference model [18]. Briefly, in the physical interference model, a transmission can be considered as successful only if the SINR at the receiver is over a given threshold. In the protocol interference model, each node i has a transmission range R_i and an interference range I_i. Node i can communicate successfully with another node j if and only if node i and j are located within each other's transmission range. When node i is transmitting to j, all other transmitters within j's interference range must be inactive. In this thesis, the term *primary interference* refers to the case when two links share the same transmitter or receiver; i.e., a node is only allowed half-duplex communication. On the other hand, *secondary interference* refers to the case where a receiver is interfered by an unintended transmitter; i.e., the transmission is destined for another receiver. Unlike the protocol or physical interference models that allow *one* transmission or reception per node, MTR capable nodes are allowed multiple transmissions *or* receptions to/from their



(a) Protocol interference model (b) MTR interference model

Figure 1.3: (a) When node B transmits to A, node C and D must be inactive. (b) Node B, C and D can transmit to A at the same time. Conversely, node A can transmit to B, D and C simultaneously.

respective neighbors; note, a node cannot transmit *and* receive at the same time. To see this, consider the topology shown in Figure 1.3a and assume nodes adhere to the protocol interference model. As nodes C and D are within node A's interference range, they must be inactive when B is transmitting to A. On the other hand, in an MTR system, node B, C and D can transmit to A concurrently; see Figure 1.3b. For these reasons, the physical and protocol model cannot be used in MTR WMNs.

The capacity of a WMN is predicated on an efficient link scheduler that ensures all transmitting links do not experience collision and nodes receive their required transmission opportunities [26]. To this end, this thesis considers spatial Time Division Multiple Access (TDMA) based link schedulers. Their key advantages include collision-free and deterministic channel access. Moreover, they provide the maximal possible network capacity, and hence, can be used as a benchmark against random access methods. A TDMA schedule or superframe consists of a number of slots. Non-interfering links are scheduled to transmit in the same time slot. Each slot may have a high number of transmitting links. Additionally, a superframe is repeated periodically. Thus, a short superframe means nodes can transmit frequently. Both cases result in high network capacity [27][28]. In addition to network capacity, other objective(s) may be considered; for example, the schedule must ensure all links have sufficient transmission opportunities to satisfy their load. Another objective is to

ensure flows receive a fair share of the network capacity.

It is worth noting that the link scheduling problem in MIMO-based MTR systems includes the sub-problem of allocating DoFs for data transmissions and interference cancellation (IC). A stream from a transmitter to an unintended receiver must be nulled/suppressed by an available DoF at either the transmitter and/or receiver side [29]. In each time slot, the total number of DoFs used by a node for transmitting /receiving data streams and IC must be less than or equal to its DoFs. Thus, a scheduler for MIMO-based MTR WMNs must aim to efficiently assign DoFs at each node such that the number of transmitting links in each slot is maximized [30].

1.2 Problem Space and Motivation

As will be shown in Chapter 2, existing link scheduling works for TDMA-based MTR WMNs only aim to maximize network capacity. Specifically, prior works do not consider the Quality of Service (QoS) of end-to-end flows, and assume each link is given a fixed traffic load. In addition, no existing works consider DoF assignment in MIMO-based MTR WMNs. Henceforth, this thesis aims to fill these critical gaps given the importance of link scheduling in MTR WMNs. It considers the following key problems: (i) optimizing the QoS of end-to-end flows, (ii) generating a robust superframe when traffic demands are random, and (iii) assigning DoFs efficiently in MIMO-based MTR WMNs with the goal of minimizing the superframe length.

1.2.1 QoS of End-to-End Flows

Minimizing end-to-end delays is important. Doing so improves the QoS of real-time multimedia applications [31]. However, guaranteeing end-to-end delays in MTR WMNs is a challenging problem. This is because one has to consider (a) the transmission order of links in the resulting superframe, (b) the superframe length, and (c) routing, which dictates the traffic load or required slots of each link.

To illustrate the problem, consider transmitting a packet on a path in a WMN.

After receiving a packet, each intermediate node has to wait for its out-going link's active time slot; this delay is called the *inter-link activation time* (IAT). Consider Figure 1.4, where links are bidirectional. Assume there is a route from node A to G through D. Loo et al.'s algorithm [32] produces the schedule shown in Table 1.2, where link AD is activated in slot 2 and link DG is activated in slot 1. Thus, the end-to-end delay of the route from A to G is $2 + 3 = 5$ slots because a packet must wait for two slots at node D before the out-going link to node G is activated; the IAT of this packet from D to G is 3 slots. On the other hand, if the slots are re-ordered, i.e., by swapping slot 1 and 2, the delay is reduced to two slots because link DG can be activated right after link AD.

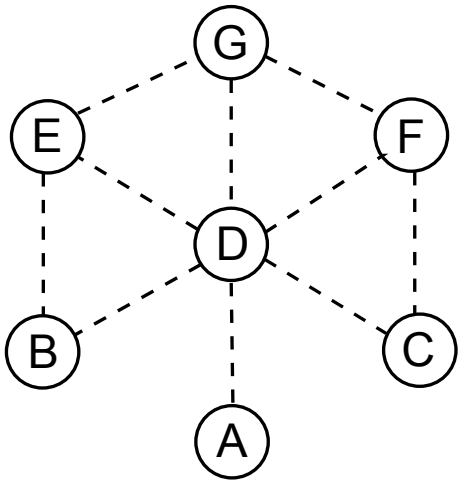


Figure 1.4: An example MTR WMN

Slot 1	**Slot 2**	**Slot 3**	**Slot 4**
DA	AD	DE	ED
DB	BD	DF	FD
DC	BE		
DG	CD		
EB	CF		
EG	GD		
FC	GE		
FG	GF		

Table 1.2: Link schedule for the topology shown in Figure 1.4

End-to-end delays are also affected by the superframe length [33], which in turn is determined by the path taken by each demand. A short superframe is preferred

because it provides frequent transmission opportunities to links such that IAT reduces if the out-going link is active before the incoming link is activated at an intermediate node. Note, the superframe length is intricately linked to the routing protocol because the number of slots required by each link is proportional to its load; i.e., a higher load requires more slots.

To illustrate the effects of routing and link load on the superframe length, consider the example topology in Figure 1.4. Assume there is only one flow from node A to G. It has two possible paths: $A-D-G$ and $A-D-E-G$. The links along each path are scheduled one after another according to Table 1.3 and 1.4. We see that the resulting superframe length as well as the end-to-end delay of these paths are two and three slots, respectively. Specifically, the additional hop in the path $A-D-E-G$ caused a longer superframe length.

Another factor that affects performance is link disjoint paths. Reconsider Figure 1.4 with the following flows: A to G, B to G and C to G. First consider the case whereby these three flows select paths $A-D-G$, $B-D-G$ and $C-D-G$, respectively. We see that link DG is used three times. In other words, link DG requires three time slots. The resultant schedule is shown in Table 1.5, which is four slots in length. Links AD, BD and CD can use one slot because of MTR. The resulting end-to-end delays of the three demands are two, three and four slots respectively. Now, consider the case whereby flows use link disjoint routing paths. That is, the flows from A to G, B to G and C to G select the following link disjoint paths: $A-D-G$, $B-E-G$ and $C-F-G$. In this case, the optimal superframe length is two, and the end-to-end delay is only two for each demand; see Table 1.6. From these examples, we see that the routing policy has a non-negligible impact on the superframe length and end-to-end delays.

Slot 1	**Slot 2**
AD	DG

Table 1.3: An example link schedule for flow from A to G

Fairness is also a key consideration whilst optimizing the QoS of end-to-end

Slot 1	Slot 2	Slot 3
AD	DE	EG

Table 1.4: Another example link schedule for flow from A to G

Slot 1	Slot 2	Slot 3	Slot 4
AD	DG		
BD		DG	
CD			DG

Table 1.5: An example schedule

flows. Schedulers that minimize the superframe length or maximize the number of links in each slot without considering fairness may starve flows. In particular, flows over multiple hops may receive less bandwidth/rate than single-hop flows. Assume flows and routing paths are given, and flows are greedy, meaning they will consume any given bandwidth. The rate allocated to a multi-hop flow is decided by the minimum number of slots assigned to the links on its path. Consider the topology in Figure 1.5. It has a multi-hop flow f_1 and a single hop flow f_2. A possible link schedule that maximizes the number of links in each slot is shown in Table 1.7. The rate allocated to f_1 is $1/3$ because the superframe length is 3 and link BC and CD are assigned with one slot, respectively. However, the rate of f_2 is $2/3$ because link DC is active in two slots. In this case, the multi-hop flow is starved. Another possible link schedule is shown in Table 1.8 that allocates $1/2$ capacity to f_1 and f_2 respectively. In this case, the two flows share the channel capacity fairly. In summary, it is challenging to ensure fairness and high network capacity. Recall that a short superframe means high network capacity. Therefore, a fundamental problem is how to generate a superframe that ensures end-to-end fairness while also minimizing the superframe length.

Slot 1	Slot 2
AD	DG
BE	EG
CF	FG

Table 1.6: The optimal schedule

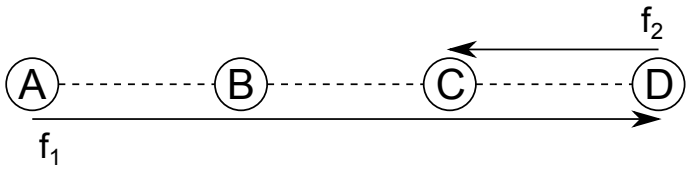


Figure 1.5: An example topology with two flows: $f_1 : A-B-C-D$ and $f_2 : D-C$

Slot 1	**Slot 2**	**Slot 3**
AB	BC	DC
CD	DC	AB

Table 1.7: An example superframe that allocates 1/3 capacity to f_1 and 2/3 capacity to f_2

1.2.2 Random Demands

A key assumption in the design of existing MTR link schedulers is that they assume the link load is fixed and known in advance. This information is then used to derive the shortest superframe. However, in practice, traffic information is likely to vary and is *random*, which may result in two critical consequences. First, when the actual traffic through a link becomes lower than when the superframe was derived, slot(s) assigned to the link will be underutilized. Second, when the load through a link becomes higher, its assigned slots will be insufficient and may lead to packets experiencing significant delays. Critically, in large scale multi-hop WMNs, it is impractical to recompute the superframe whenever the traffic load changes because this will require the scheduler to gather load information from all nodes, recompute a new schedule, and disseminate it to nodes over multiple hops, which upon installation may be dated.

When links have low loads, random channel access is preferred because a link can relinquish the channel when it has finished transmission. In contrast, for high load scenarios, a TDMA schedule is ideal as it enables links to gain channel access

Slot 1	**Slot 2**
AB	BC
CD	DC

Table 1.8: An example superframe that allocates 1/2 capacity to f_1 and f_2 respectively

at specific times in a collision free manner. Hence, a combination of TDMA and random channel access is ideal; see the superframe structure shown in Figure 1.6. One example of such a superframe can be found in wireless systems such as IEEE 802.15.3 [34]. A key limitation, however, of past works is that they assume the *boundary* between the scheduled and random access part is fixed. Ideally, if the link load is high, then the TDMA part should be sufficiently long to ensure little or no collision in the random access part. Conversely, in low load scenarios, the superframe will contain an elongated random access part to allow nodes to gain access when needed. Thus, for MTR WMNs with random link loads, a challenging problem is how to properly dimension the scheduled and random access parts in the said superframe such that the unused scheduled times and collisions are minimized.

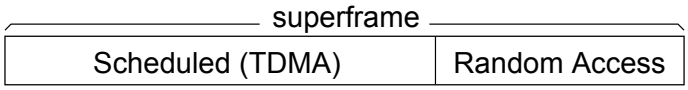


Figure 1.6: A superframe with scheduled and random access parts

Another key assumption of prior works is that the demand of end-to-end traffic is considered fixed. However, as stated in [35], it is likely to be uncertain and unknown. Consider a WMN with several end-to-end flows, and the demand of each flow is fixed. Assume each flow is given a set of available routing paths. Here, a routing can be considered as the fraction of traffic routed on each link for a given flow. An objective, such as minimizing the superframe length or minimizing end-to-end delay, can be achieved by finding a proper routing and link scheduling while considering known demands. However, if the end-to-end traffic demands are random and unpredictable, the load on each link varies. That means a link may not have a packet to transmit in its allocated slot(s), which is a waste of resource. On the other hand, the load of a link may exceed its allocated bandwidth when all flows traversing it have a high demand, which leads to severe packet delays. Moreover, uncertain demands may cause a network operator to compute and install a new routing and superframe frequently; this is likely to incur high signalling overheads, especially in large scale multi-hop WMNs. Therefore, a challenging problem is how

to find a routing and a minimal superframe length that is robust against random end-to-end demands.

1.2.3 DoF Assignment

As mentioned in Section 1.1, MIMO technology can be used to implement an MTR WMN. In particular, a node is able to transmit/receive multiple independent data streams simultaneously, so called *spatial multiplexing*. Each node is equipped with a smart antenna array and a receiver is able to separate and decode these transmissions based on the unique spatial signature of data streams. Apart from that, a subset of a node's antenna elements can be used to cancel interference to/from neighboring links in order to increase spatial reuse; the process of determining the number of antenna elements or DoFs for transmission and IC is also known as *stream control* [36]. A transmitter's data streams that cause interference to a neighbor or an un-intended receiver are called *interfering streams*.

Consider Figure 1.7. Node A is transmitting one data stream to node B, and node C is transmitting two data streams to node D. Node C has two interfering streams from node B. If node B wants to suppress the interference streams from node C, node B must assign two DoFs for IC. If node C wants to null the interference to node B, it must assign one DoF for IC because node A has one data stream to node B. Recall that at each node, the number of DoFs used for data transmission and IC must be less than or equal to the node's total available DoFs. Thus, the DoF assignment at each node will affect the number of data streams in each time slot. Therefore, an open question is whether different DoF assignment strategies have an impact on on the superframe length of MIMO-based MTR WMNs.

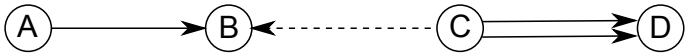


Figure 1.7: An interference example

1.3 Contributions

This thesis addresses the foregone problems and outlines a number of novel solutions/algorithms; see Table 1.9. Specifically, it contains the following contributions.

1.3.1 Joint Routing and Scheduling Algorithms for Minimizing End-to-End Delays

The goal is to design an approach that jointly considers routing and scheduling to address two fundamental issues that influence end-to-end delays: superframe length and transmission slot order. Shortening the superframe length, in terms of slots, is expected to minimize the IAT whilst reordering transmission slots increases the likelihood that links on a path are activated consecutively. This thesis proposes two algorithms. The first called JRS-Multi-DEC uses a novel metric to minimize the load of each link whilst the second, called JRS-BIP, uses a binary integer program (BIP) approach. Both algorithms aim to minimize the overall delay and use slot re-ordering on the resulting schedule to further reduce delay. Numerical results show both algorithms are able to reduce the average end-to-end delay by approximately 50% as compared to a non joint routing algorithm. This thesis also analyzes the relationship between superframe length, routing paths, link weights and end-to-end delay. In addition, it proves that re-ordering slots reduces end-to-end delay by at most $H(|S|-2)+1$ slots for a demand with H hops and superframe length of $|S|$.

1.3.2 Flow Aware Fair Scheduler

Recall that no existing works for MTR WMNs consider flow rate nor any notion of fairness when deriving a TDMA schedule. This thesis addresses this critical gap by proposing a link scheduler, called Algo-Fair; it jointly maximizes network capacity and also considers flow rates fairness. Further, it uses a novel augmentation step to distribute spare capacity fairly amongst flows. This step is general and can be applied readily in other forms of wireless networks. Apart from that, Algo-Fair

generates a schedule directly whilst yielding a fair rate allocation. This is different from existing methods that first use a flow contention graph to compute a fair share before deriving the corresponding schedule, which may not exist. Numerical results show that Algo-Fair is able to achieve the optimal Max-min Fair (MMF) allocation in some cases, and on average, has a gap of 3.7% to the optimal solution. Moreover, Algo-Fair is able to generate a higher minimum flow rate as well as a higher average throughput than competing approaches.

1.3.3 Superframe Construction for Wireless Networks with Stochastic Demands

As mentioned, in practice, link load is likely to vary, meaning the computed schedule or superframe will have unnecessary idle times. To this end, this thesis proposes a stochastic programming (SP) based approach to generate a superframe comprising of a TDMA and a random access part. Advantageously, it sizes both parts according to traffic distribution. This approach avoids recomputing a new superframe whenever the traffic demands change. In addition, this SP approach has a *control knob*, in the form of a penalty value, that allows a network operator to size both parts of the derived superframe according to the offered traffic load. Moreover, this approach can be embedded within a binary search to obtain the best penalty value for a given probability of collision and idle times. Lastly, this approach is general and can be used for example to size the contention access period (CAP) and channel time allocation period (CTAP) of IEEE 802.15.3 wireless networks [34]; similarly, it can be used in other forms of wireless networks.

1.3.4 Joint Routing and Scheduling with Random Demands

Uncertain end-to-end demands may cause a network operator to compute and install a new routing and superframe frequently; this is likely to incur high signaling overheads, especially in large scale multi-hop WMNs. Henceforth, this thesis considers

Problems	**Algorithms**	**Cross Layer**	**Algorithm Type**	**Comments**
End-to-End Delay	JRS-Multi-DEC	Routing and Scheduling	Heuristic	
	JRS-BIP	Routing and Scheduling	BIP and Heuristic	
Flow Aware Fairness	Algo-Fair	Scheduling	Heuristic	Fairness and Network Capacity
Random Link Load	SP-based Approach	Scheduling	SP	Superframe construction
Random End-to-End Demand	Algo-PolyH	Routing and Scheduling	LP and Heuristic	Polyhedral model
DoF Assignment	Algo-MIMO	Scheduling	Heuristic	Stream Control

Table 1.9: A list of contributions

random traffic demands characterized by a polyhedral set, and models the problem as a semi-infinite Linear Program (LP). A novel algorithm, called Algo-PolyH, is then proposed. It jointly considers both routing and superframe generation to produce a *robust* solution that is valid for all random demands that belong to a given polyhedral set. This is confirmed in experiments over networks with varying node degrees, number of flows, number of nodes and number of paths.

1.3.5 An Efficient Link Scheduler for MIMO WMNs

The DoFs or antenna elements available at each node enable concurrent transmission/reception of independent data streams. Alternatively, they can be used to suppress interference. This thesis studies the problem of minimizing the TDMA superframe length, in terms of slots, of an MTR MIMO-based WMN. A novel heuristic algorithm named Algo-MIMO is proposed. It uses a recently proposed node ordering DoF model for Interference Cancellation (IC). Numerical results show that Algo-MIMO is able to reduce the superframe length by up to 60% as compared to algorithms that use other IC models, and approximately 40% against algorithms that order nodes based on the number of neighbors, node weight or randomly.

1.4 Publications

1. **L.Y Wang**, K-W Chin, S. Soh, and R. Raad. *Delay-Aware Joint Routing and Scheduling for Multi Tx-Rx Wireless Mesh Networks*, IEEE International Conference on Communications (ICC), Sydney, Australia, June, 2014.

2. **L.Y Wang**, K-W Chin and S. Soh. *An Efficient Link Scheduler for MIMO Wireless Mesh Networks*, IEEE Second International Workshop on Ad Hoc Networking with MIMO and Cognitive Radio, London, United Kingdom, June, 2015.

3. **L.Y 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*, Elsevier Computer Communications, 2015. *Accepted, available online.*

4. **L.Y Wang**, K-W Chin and S. Soh. *On Constructing Superframes for Wireless Networks with Stochastic Demands*, IEEE Communications Letters, 19(10), October, 2015. *To appear, available online.*

5. **L.Y Wang**, K-W Chin and S. Soh.*A Novel Fair Link Scheduler for Multi Tx/Rx Wireless Mesh Networks.* Elsevier Computer Networks. *Under Review*

6. **L.Y Wang**, K-W Chin and S. Soh. *Joint Routing and Scheduling in MTR WMNs with Random Demands*, Elsevier Computer Networks. *Under Review*

1.5 Thesis Structure

1. *Chapter 2.* It contains a survey of related works on joint routing and link scheduling problems in TDMA-based WMNs with fixed or random traffic demands.

2. *Chapter 3.* It outlines two algorithms that jointly optimize end-to-end delay and link schedule.

3. *Chapter 4.* This chapter proposes Algo-Fair. Its aim is to derive a schedule that allocates flow rates fairly using a novel augmentation step.

4. *Chapter 5.* This chapter presents a stochastic programming based algorithm to dimension a hybrid superframe containing a scheduled and random access part.

5. *Chapter 6.* This chapter considers random demands characterized by a polyhedra model and proposes a joint routing and scheduling algorithm called Algo-PolyH.

6. *Chapter 7.* This chapter considers DoF assignment and schedule generation. It studies different assignment methods and outlines an algorithm called Algo-MIMO.

7. *Chapter 8.* This chapter concludes the thesis, and provides a summary of key contributions and presents future research directions.

Chapter 2

Literature Review

This chapter reviews prior works that consider joint routing and scheduling, fair scheduling, random demands and DoF assignment for TDMA-based WMNs. As we will see, these works either do not consider MTR systems, or the problems discussed in Section 1.2. A summary of their limitations will be presented in Section 2.5.

2.1 Joint Routing and Scheduling

A routing protocol is responsible for selecting one or more paths for each traffic flow. Hence, the link load has a direct relationship with the chosen routes. After that, a scheduler or Medium Access Control (MAC) protocol assigns the required slots to each link in order to satisfy its load. Moreover, it may decide the transmission order of links with the goal of minimizing end-to-end delays. Hence, in the past few years, there have been a number of joint routing and scheduling (JRS) approaches. The key challenge is taking into account interference, which is determined by antenna type as well as simultaneously transmitting links.

To date, researchers have considered many antenna types. They include omni [31], directional [21], smart [27] and beam-forming [37] antennas. Each antenna type has different characteristics and interference footprint. Consequently, they have a significant impact on the set of transmitting links. This section thus groups

JRS approaches according to their assumption on antenna type. Specifically, Section 2.1.1 discusses works that use omni-directional antennas; i.e., the signals from nodes propagate in all directions. After that, Section 2.1.2 outlines works that employ directional, smart or beam-forming antennas; i.e., nodes are able to focus their transmission in a given direction or have the ability to null interference.

2.1.1 Omni-Directional

Nodes with an omni-directional antenna are allowed one transmission or reception at a given time. Moreover, transmissions may cause interference to neighboring nodes. Section 2.1.1.1 summarizes works that consider a single channel, and Section 2.1.1.2 outlines those that employ multiple channels to avoid interference. As will be discussed later, the major problems addressed by these works include (i) maximizing network throughput, (ii) deriving the minimum schedule length, (iii) minimizing end-to-end delays, and (iv) delivering packets within a deadline.

2.1.1.1 Single Channel

A key problem addressed by works that consider single-channel TDMA-based WMNs is constructing an interference aware routing tree that results in a short schedule. Another problem is minimizing end-to-end delays subject to interference constraints. Lastly, researchers have also addressed the problem of determining routing paths and link activation patterns such that packets are delivered before their expiration time.

The routing tree plays a critical role in determining the interference experienced by a flow. To this end, the authors of [38] aim to construct an interference aware routing tree that leads to the minimum schedule length. In particular, they assume the physical interference model, whereby a transmission is successful only if the signal-to-interference-plus-noise ratio (SINR) at the receiver is higher than a predetermined threshold [18]. The links that form the routing tree are determined using a Mixed Integer Linear Program (MILP). The aim is to select a set of links that

can be activated using a short schedule. They showed that the MILP formulation is NP-complete. They then propose an iterative pruning-based routing algorithm that considers link scheduling when constructing a spanning tree. The key idea is to remove links that cause severe interference so that more links can be active in the same slot. In addition, the authors consider transmission power control to reduce interference and improve spatial reuse. Their algorithm runs Dijkstra's algorithm with transmission power as link cost. Least cost paths thus correspond to those that use the minimum transmission power and have the least interference. However, these paths do not lead to the shortest frame length if they include links that produce high interference to neighboring links. To this end, the algorithm iteratively removes these links while ensuring connectivity and reruns Dijkstra's shortest-path algorithm until the number of iterations meets a given threshold.

The work in [39] proposes a Network Utility Maximization [40] framework that jointly determines the rate and link schedule of unicast and multicast sessions. In addition, the framework guarantees end-to-end throughput and bounds the delay of traffic demands. They assume the routing tree is given. They formulate the problem as a convex optimization problem, and then solve its Lagrange dual. The results are the optimal rate vectors for each session. Next, they propose a link scheduler that employs graph coloring; links assigned the same color are scheduled in the same slot. The number of slots in the resulting schedule is equal to the required number of colors. The key idea is to transform the conflict graph into a Chordal graph and make use of the known fact that Chordal graphs can be colored optimally using a greedy coloring algorithm [41]. To this end, they use the LEX M algorithm [42] to transform the conflict graph for a given topology into a Chordal graph. For each node, a greedy coloring algorithm assigns the smallest color that is not used by its neighbors. The generated coloring result is the shortest schedule.

The work in [33] jointly considers minimizing end-to-end delay and schedule length. The authors showed that scheduling delay occurs when an outbound link is scheduled before an inbound link. Thus, end-to-end delays and schedule length are

affected by transmission duration, activation time and transmission order of links. The authors also showed that the problem is similar to the NP-complete Periodic Event Scheduling Problem (PESP) [43] with the constraint that no two interfering links overlap in time. The authors model the scheduling problem as an Integer Linear Program (ILP) with the objective of finding the activation time of each link and transmission order of links. To solve PESP, the authors propose a polynomial time algorithm that uses the Bellman-Ford algorithm and assume the transmission order is fixed. They first generate a conflict graph. The key idea is to use the delay between two links as the arc costs in the conflict graph so that the shortest path tree found by the Bellman-Ford algorithm is the minimum schedule. Then, they propose a polynomial time algorithm to determine the transmission order for paths on tree topologies. Ideally, the transmission order, including those on the return path, should be consecutive. To that end, given a tree topology, they assign a rank to links based on their distance from the root of the tree. A link with a smaller rank will transmit first. This transmission order is then used as an input to the Bellman-Ford algorithm.

Unlike [38], [39] and [33], the authors of [31] consider JRS and end-to-end delays. They propose a cross-layer algorithm that computes routing and scheduling separately. The authors propose a LP to find paths with minimal interference. The LP includes flow conservation and link capacity constraints. In addition to routing, the LP also yields the link load. The aim of the scheduling part is to decide the activation time of each link and transmission orders so that the total end-to-end delay is minimized. They propose an ILP with the objective of minimizing the total delay. The binary decision variables indicate whether link l is active in slot t. However, the ILP is shown to be NP-hard. To solve it, the authors firstly relax it into a non-integer LP and round the derived solutions into integers. Specifically, they set a rounding threshold of 0.5. In other words, if a fractional solution is above 0.5, it is rounded to one or it is active in a given slot. Otherwise, it will be rounded to zero.

The JRS approach in [44] aims to minimize the end-to-end worst-case delay

(WCD) and deadline. This approach is based on an Mixed Integer Non-Linear Program (MINLP). The decision variables decide whether a link is used by a flow, its activation time, transmission duration, and transmission order. The objective is to minimize the maximum deadline violation of each flow, which is equal to the difference between a flow's WCD and its deadline. The MINLP formulation considers the following constraints: (i) interfering links must not be scheduled in the same slot, (ii) the number of slots assigned to each link must be less than the peak rate and the burst size of flows, and (iii) flow conservation constraints. The authors showed that the resulting MINLP is solvable only for a 4x4 grid (16 nodes). To improve computation speed, they first determine the transmission order of links before solving the MINLP; i.e., the transmission order is no longer a decision variable in the MINLP. This is similar to Djukic et al.'s approach [33], whereby the transmission order of a link corresponds to its rank. The rank of a link is determined by its distance to the root node. In [44], they decide the transmission order using a conflict graph and by employing a general K-coloring method [45] to minimize the maximum number of colors. Then, they use Lagrangian relaxation to decouple the link scheduling and routing constraints from the MINLP. This yields two smaller problems which can be solved separately by any LP solver.

The work in [46] considers JRS and packet deadline. The authors first use an ILP to find a feasible link schedule. The decision variables indicate whether a link is active in a slot. However, this ILP formulation is shown to be NP-complete. Thus, the authors propose a genetic approach (GA) to find a feasible schedule. The main idea is to generate the best individuals from an initial population. Each individual is a possible schedule. Each schedule includes a string of gene values. Each gene value denotes whether node n is transmitting to node k in time slot t. GA aims to find a combination of gene values (schedule) that delivers packets within a given deadline. Each iteration creates a new generation of individuals (link schedules). GA first applies stochastic universal sampling [47] to choose parents. Then, it generates a new generation by randomly picking the gene values of both parents. To ensure

diversity, GA will randomly change the gene of individuals with uniform probability. In each iteration, a fitness function considers the number of activated links per slot and the delay of each route. A schedule is feasible if it can deliver all traffic without violating interference and deadline constraints. Also, the fitness function will give a feasible schedule a higher value if it yields routing paths that are shorter. The final solution is the schedule with the highest value.

Table 2.1 summarizes the aforementioned works. We see that references [33] and [39] only focus on the MAC layer. In other words, they have not considered JRS. The works in [39], [33] and [44] use a conflict graph, which is computationally expensive. The authors of [38], [39] and [33] only consider tree topologies, and their applicability in general topologies is an open question. In terms of the interference model, we see that most works employ the protocol interference model as opposed to the physical model. The main advantage of the protocol interference model is its idealistic property and thus simplifies analysis. On the other hand, the physical interference model better reflects reality as it considers the cumulative interference caused by nearby transmissions [18].

2.1.1.2 Multiple Channels

In this section, all works consider a multi-channel TDMA-based WMN where each node is equipped with one or more omni-directional antennas. Moreover, multiple orthogonal channels are used to reduce interference and increase the number of concurrent transmissions. A key challenge is that in practice there are a limited number of channels. Another key problem is ensuring each channel contains the minimal number of interfering links. That is, when deriving a schedule, in each time slot, a scheduler has to determine interfering links and assign them to orthogonal channels.

In [48], the authors assume there are sufficient orthogonal channels to assign to interfering links. The authors first formulate the problem as an LP to determine whether a link is included in the routing tree, the activation time of links and their

Approaches	**Objectives**	**JRS**	**Tree**	**Interference Model**	**Formulation**	**Solution**
Friderikos et al. [38]	Minimize schedule length	No	Yes	Physical	MILP	Iterative pruning-based routing algorithm
Wang et al. [39]	Guarantee throughput and minimize schedule length	No	Yes	Protocol	N/A	Graph-coloring based scheduling
Djukic et al. [33]	Minimize delay and schedule length	No	Yes	Protocol	PESP	Bellman-Ford based algorithm
Cheng et al. [31]	Minimize delay	Yes	No	Protocol	ILP	Relax to LP and round fractional solutions
Cappanera et al. [44]	Minimize delay and schedule length	Yes	No	Protocol	MINLP	Graph-coloring and Lagrangian decomposition
Badia et al. [46]	Deliver packets within deadline	Yes	No	Physical	ILP	Genetic approach

Table 2.1: A comparison of routing and scheduling works that employ single-channel and an omni-directional antenna

corresponding time slots. However, with increasing network size, the number of link configurations increases exponentially. Hence, the authors propose a heuristic algorithm with the following two phases: (i) given a topology and a set of flows, construct a routing tree that results in the shortest schedule length, and (ii) given a routing tree and traffic requirements, schedule links in the minimum number of slots. In phase (i), at each node, the number of required slots depends on its incident links. The authors define the weight of a link as the reciprocal of the link capacity. The weight of a node is the total weight of all its incident links. The authors then apply a modified Dijkstra's shortest path algorithm from the gateway to each destination. The path cost is defined as the maximum node weight along the path. For each flow, the algorithm finds the optimal path that needs the minimal number of slots. Then, the path with the highest cost is added to the routing tree. After obtaining one path, the algorithm updates all link and node weights in accordance with link loads. It then re-constructs the next path using the same process. It repeats this procedure until all paths are added. The generated routing tree is then used in phase (ii). Specifically, they construct a conflict graph for the tree, which is then transformed into a Chordal graph using the technique in [39]. Lastly, the Chordal graph is colored using a greedy algorithm to yield the shortest schedule.

The work in [49] first defines a concurrent transmission pattern (CTP) as a set of links that can be active at the same time over multiple channels whilst satisfying half-duplex and radio constraints. The radio constraint restricts the maximum number of concurrent communications to be the number of radios at each node. The authors then propose an LP with the objective of minimizing the schedule length. The decision variables are routing variables that determine the amount of traffic routed on each link, and scheduling variables that decide the activation time of each CTP. However, the LP becomes intractable with increasing number of nodes, links and channels. This is because the number of CTPs increases exponentially. To reduce computational complexity, the authors propose a column generation [50] based approach that decomposes the problem into a master problem and a sub-problem

that can be solved by any LP solver. The master problem is the proposed LP with a set of feasible CTPs. The sub-problem is to find additional CTPs that reduce the schedule length. The proposed approach stops when it finds the set of CTPs, routing paths and link activation times that lead to the shortest schedule.

The authors of [51] consider JRS in WiMAX-based WMNs. The authors first define a transmission group as a set of links that can be active at the same time. Then, they propose a link-based ILP with the objective of minimizing the schedule length. The aim is to construct a routing tree, determine link loads and decide the activation time of each transmission group. To reduce the number of routing decision variables, they propose a path-based ILP whereby the routing paths are given and the goal is to select the paths used to form the routing tree and generate the shortest schedule. However, the number of transmission groups increases exponentially with the network size. To this end, the authors use column generation to decompose the problem into a master and a sub-problem. The master problem is simply the relaxed version of the proposed ILP. The sub-problem is to find transmission groups. The resulting solution includes transmission groups, their activation times and a routing tree that minimizes the total schedule length.

Different from the previous two works, the authors of [52] divide the problem into two sub-problems: (i) find the best routing paths and the activation time of a set of concurrent transmission configurations such that the schedule length is minimum, and (ii) how to assign transmission configurations to orthogonal channels to further reduce schedule length. In (i), they present an MILP formulation to determine routing paths and link schedule in single channel scenarios. The authors employ column generation to generate feasible configurations. Each configuration is a set of links that can be active in the same slot. To further reduce the schedule length, in (ii), they propose an ILP with the objective of minimizing the number of required time slots to determine whether a configuration is assigned to a slot over a given channel. They consider the following constraints: 1) each configuration can only be assigned to one slot, 2) in each slot, the number of used channels cannot exceed

the maximal number of available orthogonal channels, and 3) a node does not have more active communications than the available number of interfaces.

The work in [53] considers the end-to-end delay of voice traffic. The authors aim to schedule the maximum number of voice calls admitted in a given period subject to delay constraint. They propose an on-line delay-constrained heuristic algorithm called DelayCheck. It finds a routing path, channel assignment and link schedule for each new coming call. It first constructs an auxiliary graph in which each vertex is a tuple (v, s, c, d), where v is the node ID, s is the slot number, c is the channel number and the delay d is calculated from the source of the incoming call to v. Each vertex (tuple) can be considered as a possible combination of routing, channel assignment and link scheduling for a link. There will be an edge between two vertices, e.g., (v, s, c, d) and (v', s', c', d'), if the corresponding two nodes v and v' are neighbors and their channel assignment and link scheduling will not lead to collision. With the auxiliary graph in hand, the authors then run Dijkstra's shortest path algorithm so that the tuples along the selected path provide a feasible routing path, channel assignment and a link schedule that satisfy the delay constraint of a new call. In addition, as a comparison, the authors formulate the off-line delay-constrained joint problem as an ILP with the objective of maximizing the number of calls accepted. Numerical results show that DelayCheck is able to accept 93% of voice calls as compared to this upper bound.

In a similar work, Shetiya et al. [54] consider the problem of providing Quality of Service (QoS) to real and non-real time traffic through JRS in WiMAX-based WMNs. They first propose a routing algorithm for both real-time and best-effort applications. The key idea is to use Dijkstra's or Bellman-Ford's shortest path algorithm to construct a routing tree whereby the link cost is set to the reciprocal of the assigned transmission rate. The derived routing tree is used as an input in the link scheduling stage. A separate scheduling algorithm is used for UDP and TCP traffic because they have different QoS requirements. For UDP traffic, the authors propose an LP to find the minimal number of slots required by each link in the final

routing tree. The scheduling result must ensure that a given amount of traffic can be delivered to the destination within a period of time and also, the end-to-end drop probability is bounded. As for TCP traffic, they first assign a fixed number of slots per frame to each node based on its average data arrival rate and the estimated channel capacity. However, in each frame, several links may not have sufficient data to fully utilize their assigned slots. Thus, slots are assigned to nodes with sufficient data first. To ensure nodes receive the required throughput in the long term, each node is associated with a counter that records the number of assigned slots. When the counter reaches an upper bound, slots will be assigned to other nodes.

In [55], a distributed protocol called Joint Multi-channel and Multi-path control (JMM) is proposed to increase end-to-end throughput. The main approach is to perform joint multi-channel link scheduling and multi-path routing [55]. Each node knows the channel information of its two-hop neighbors and thereby allowing it to select the least used channel. A transmitter switches to its receiver's channel before transmission. As for routing, the authors consider the problem of constructing two link-disjoint routing paths from each node to its gateway to increase end-to-end throughput. The key idea is that each node broadcasts a packet that records all nodes and the number of hops to its gateway. The gateway then selects a pair of node-disjoint paths with similar number of hops to a node. In terms of scheduling, each node needs to decide to transmit or receive in each time slot to ensure packets can be delivered successfully to destinations. Recall that a fixed number of slots are organized into a superframe. The authors simplify the superframe at each node as two continuous parts. In each part, a node can be transmitting or receiving, but not both. Then, the problem is to decide the modes, i.e., transmitting or receiving, of these two parts to avoid collision. Given a routing path, a node decides its mode based on the action of its neighboring nodes. For example, if one of node i's neighbors is transmitting in the first half and receiving in the second half of the superframe, then node i will receive in the first half and transmit in the second half.

Table 2.2 summarizes the aforementioned works. All works jointly consider rout-

Approaches	Objectives	Tree	Interference Model	Formulation	Solution
Wang et al. [48]	Minimize schedule length	Yes	Protocol	LP	Modified Dijkstra's algorithm
Zhang et al. [49]	Minimize schedule length	No	Protocol	LP	Column generation
El-Najjar et al. [51]	Minimize schedule length	Yes	Physical	ILP	Column generation
Capone et al. [52]	Minimize schedule length	No	Physical	MILP and ILP	Column generation
Gabale et al. [53]	Provide delay guarantee	No	Protocol	ILP	Construct an auxiliary graph and run Dijkstra's algorithm
Shetiya et al. [54]	Ensure QoS	Yes	Protocol	LP	Assign slots to nodes with higher loads first
Tam et al. [55]	Maximize end-to-end throughput	No	Protocol	N/A	Construct two link-disjoint paths and nodes decide channels and superframe based on neighbors

Table 2.2: A comparison of JRS works that employ multiple radios and multiple channels

ing, link scheduling and channel assignment. However, references [48] and [54] assume pre-assigned channels. References [51] [48] and [54] assume a tree topology. Only the works in [52] and [51] employ the physical interference model, whilst others consider the protocol model. We also see that column generation is a popular method adopted by past works.

2.1.2 Directional, Smart or Beam-forming Antennas

The nodes in WMNs can be equipped with one or more directional, smart or beam-forming antennas [1]. A directional antenna is able to focus electromagnetic energy onto a specific region to yield a high transmission gain, and thereby improve reception quality. Moreover, it reduces interference with neighboring links. Consequently, the use of directional antennas results in higher spatial reuse as compared to omni-

directional antennas. To this end, Section 2.1.2.1 summarizes works that employ steerable and switched beam antennas. Section 2.1.2.2 outlines works that use smart or single beam-forming antennas. Lastly, Section 2.1.2.3 discusses works that employ a Multiple-Input Multiple-Output (MIMO) system. The major problems addressed in these works include: (i) maximizing network throughput, (ii) maximizing end-to-end throughput, (iii) how to achieve the minimum schedule length, and (iv) minimizing end-to-end delays.

2.1.2.1 Conventional Directional Antennas

Unlike nodes with a single radio, those with multiple radios can perform multi-transmit-receive (MTR) or multiple transmissions and receptions. Section 2.1.2.1 summarizes works that employ a single steerable antenna. Section 2.1.2.1 outlines works that use multiple switched beam antennas.

Single Radio

The problem of computing a conflict-free schedule that guarantees all end-to-end delays are within a given bound is addressed in [56]. In this work, the delay of a flow is a function of queuing and scheduling delays. The queuing delay is determined by the number of flows sharing the same outgoing link. The scheduling delay is affected by the transmission order of outgoing and incoming links. The authors of [56] formulate an optimization problem that aims to find the minimal frame length, activation time for each link and the duration of each transmission subject to a delay constraint. However, the problem has a high computational complexity because of integer decision variables that represent frame length and number of slots for each link. Moreover, the resulting formulation is non-linear and non-differential. Hence, the authors propose an iterative approach that separately considers scheduling and queuing delay. In each iteration, the approach has two stages: (i) formulate and solve an ILP to compute the end-to-end delay for each flow, and (ii) select flows with end-to-end delay that is higher than a given threshold. In (i), the ILP is used

to obtain the minimum schedule length and scheduling delay. In (ii), the algorithm aims to reduce the delay of selected flows so that the total delay is lower than a given threshold. To reduce the queuing delay of selected flows, in the next iteration, these flows are assigned a higher weight in the ILP to obtain more slots. These two stages are repeated until the delay of all flows are within a given bound.

Similar to [56], the authors of [57] only consider primary interference. They address the problem of finding a feasible flow arrangement such that the maximal number of links are activated in each slot. They propose an algorithm called minimum consumption routing and scheduling (MCRS). It aims to find the optimal flow arrangement that blocks the fewest links and uses the least slots in the resulting schedule. Given the traffic demand of each flow, its key idea is to run the Bellman-Ford algorithm to find the shortest path, where link cost is defined as the total number of interfering links and required slots. In addition, the links on the shortest path found will experience minimal interference. Lastly, links are assigned slots to meet their load.

The authors of [28] address the JRS problem over the physical interference model. They first define a configuration as a set of links that can be active at the same time; i.e., these links satisfy SINR and half-duplex constraint. Then, they propose an MILP with the objective to minimize schedule length. In particular, the MILP has as decision variables the number of slots to be allocated to each configuration and routing paths. To evaluate the effects of antenna type, power control and rate adaptation, they propose three MILPs. First, they consider transmission power and data rate are fixed. Then, the decision variables are the amount of traffic routed over each link, the activation time of each configuration and whether a link is active in a configuration. Second, the authors consider variable transmission power with a fixed data rate. Third, the authors consider both transmission power and data rate are decision variables. Each link is able to select a data rate with a specific transmission power. Thus, a configuration includes a set of non-interfering links and data rate. These MILPs are solved using column generation.

Multiple Radios

Nodes can be equipped with multiple directional antennas to enable so called Multi-Transmit/Receive (MTR) communications; i.e., these nodes can concurrently transmit to or receive from multiple neighbors at the same time. However, nodes do not have full-duplex capability. An example MTR WMN is proposed by Raman et al. [21]. This work aims to create a low cost, long distance WMN that interconnects rural villages in India. In particular, it makes use of off-the-shelf IEEE 802.11 hardware and parabolic antennas to create mesh routers capable of transmitting or receiving simultaneously. To take advantage of the concurrent transmit or receive capability of nodes, they presented a Spatial reuse Time Division Multiple Access (STDMA) scheduling protocol. The protocol, called 2P, operates in a centralized, synchronous manner whereby routers switch between two phases: Synchronous Transmitting (SynTx) and Synchronous Receiving (SynRx). This means when a node is in SynTx, all its links are transmitting.

In [58], Kodialam et al. assume a node can receive packets from multiple neighbors via orthogonal channels and transmit to at most one neighbor at any given time. They first study the problem of determining whether a given set of end-to-end flow rates are achievable. For half-duplex nodes, they show that the total fraction of activation time must be less than 2/3. For full-duplex nodes, the total fraction of activation time must be less than one. These facts are then used in a standard maximum flow LP formulation to obtain the achievable flow rates in a given WMN. To solve the LP, they develop a fully polynomial time approximation scheme (FPTAS) primal-dual algorithm that iteratively updates the dual variables of the LP until the primal problem is feasible.

The authors of [59] propose a new channel allocation scheme, called Directed Edge Coloring (DEC), for MTR WMNs. The goal is to assign outgoing and incoming links to non-interfering channels so that a node can transmit and receive at the same time. They consider the channel assignment problem as an edge coloring problem.

The aim is to assign a color to nodes using the minimum number of colors. Given a bidirectional graph G_d, the algorithm first generates the corresponding undirected graph and colors the vertices using k colors. They then calculate the minimum n that satisfies $\binom{n}{\lfloor \frac{n}{2} \rfloor} \geq k$. Here, the value of n is the minimum number of colors used in G_d. The authors then show how to color edges with n colors. With the undirected graph in hand, nodes with the same color can be considered as a node set. It then assigns a subset of n colors of size $\lfloor \frac{n}{2} \rfloor$ to each node set. Note that no color subset is contained in another. In graph G_d, two neighboring nodes A and B must be included in different node sets with different color subsets. Then, the directed link from A to B are colored with the color that is assigned to A, but not B. Similarly, the link from B to A is colored with the color that is only assigned to B. Thus, for each node, the outgoing and incoming links are assigned with a different color. The total number of colors is constrained by n. The coloring result is a feasible channel assignment and the links with same color are assigned to the same channel.

Based on the work of [58] and [21], Dutta et al. consider the maximum concurrent flow problem [60] in MTR WMNs. They propose a JRS scheme. They first use an LP to obtain the maximum concurrent flows of each demand. However, the LP requires all possible bipartite graphs, which is exponential in number. Hence, they propose a novel scheduling algorithm, called Multi-DEC, to color the edges with the minimum number of colors for a given topology and link loads. Given a directed graph, the scheduling algorithm first calculates the weight $w(e)$ of link e based on the total flow traversing e and link e's capacity. Then, to construct a multi-graph, a weighted link e is replaced with $w(e)$ parallel directed links where each link has a weight of one. The authors aim to color the multi-graph subject to the following two constraints: 1) outgoing and incoming links have a distinct color, and 2) no two parallel directed links are allocated with the same color. Constraint 1) ensures a node transmits and receives using different radios. Constraint 2) limits a link to transmit at most one packet to one neighbor. To satisfy constraint 2), the algorithm splits the multi-graph into several simple sub-graphs in which there is only one directed link with a weight

of one between every two nodes. To satisfy constraint 1), each simple sub-graph is colored using the DEC algorithm proposed in [59].

The problem of maximizing network throughput in 60 GHz WMNs subject to half-duplex and radio constraints is addressed in [61]. A key characteristic of the 60 GHz frequency band is that links can be considered as pseudo-wires. Consequently, there is no secondary interference. The main constraint is the number of links incident on a node must be less than its available radios or its neighbors. The authors divide the problem into two stages: (i) finding the maximal link loads subject to the main constraint, and (ii) assigning links to channels such that the resulting schedule uses the fewest slots possible; i.e., the assignment maximizes network throughput. In (i), the authors employ an LP to compute the flow load of each link. The resultant routing paths and link loads are inputs of (ii). In (ii), the authors propose a link scheduling and channel assignment algorithm. The key idea is to assign the channel with the highest capacity to the link with the heaviest load.

Table 2.3 summarizes the aforementioned works. Only [28] considers the physical interference model. Cappanera et al. [56], Dutta et al. [60] and Lu et al. [57] assume the side lobe of directional antennas do not cause interference to neighboring links, which is not realistic. The authors of [56] assume a tree topology. A key observation is that MTR is able to provide higher network throughput. Consequently, research into MTR WMNs is a promising direction. Although the use of multiple channels yields better capacity, it is at the expense of computational cost. Moreover, there are insufficient orthogonal channels to ensure collision-free transmissions.

2.1.2.2 Smart and Beam-forming Antennas

This section reviews works that consider the different capabilities of smart antennas. For example, the use of beam-forming (BF) to suppress interference, Spatial Division Multiple Access (SDMA) to enable concurrent transmissions between nodes, and Spatial Division Multiplexing (SDM) to provide high data rates. Interestingly, nodes with one or more beam-forming antennas are capable of multiple packets reception

Approaches	**Objectives**	**JRS**	**Transmission Mode**	**Formulation**	**Solution**
Cappanera et al. [56]	Provide delay guarantees	No	Single transmission or reception	ILP	Consider scheduling and queuing delay separately
Lu et al. [57]	Ensure end-to-end bandwidth allocation	Yes	Single transmission or reception	N/A	Modified Bellman-Ford algorithm
Capone et al. [28]	Minimize schedule length	Yes	Single transmission or reception	MILP	Column generation
Kodialam et al. [58]	Maximize end-to-end throughput	Yes	Single transmission or/and multiple reception	LP	Iteratively select the shortest path for each flow
Dutta et al. [59]	Maximize network throughput	No	MTR	N/A	Construct corresponding undirected graph and run modified graph-coloring
Dutta et al. [60]	Maximize network throughput	Yes	MTR	LP	Construct multi-graph and run modified graph-coloring
Su et al. [61]	Maximize network throughput	Yes	MTR	LP	Assign a channel to the link with the highest load

Table 2.3: A comparison of works employing directional antennas

(MPR). However, each antenna is only allowed transmission or reception to/from at most one neighbor; i.e., nodes do not have full-duplex capability. Also, each node can only receive packets from a finite number of neighbors at any given time; aka MPR reception constraint. The number of neighbors corresponds to the available antennas at each node.

The work in [27] aims to maximize network throughput. They consider three different physical layer assumptions: BF, BF+SDMA, and BF+SDMA+SDM. In the first case, all antennas are used to suppress interference. In the second case, antennas are employed to suppress interference or to transmit (receive) to (from) multiple nodes. In the last case, antennas are used to provide interference suppression, concurrent transmissions/receptions or higher data rates. The authors propose three JRS LPs with the objective of maximizing network throughput to decide routing paths and schedule. They consider the flow conservation, link capacity and radio constraints. To solve these LPs, the authors use column generation to iteratively find link transmission sets that can improve throughput. The generated results are routing paths, traffic load on each link, link transmission sets and the activation time of each transmission set.

Both [62] and [63] consider MPR capable nodes and aim to reduce interference in order to maximize network throughput via joint routing and scheduling. They consider an MPR protocol model whereby at any time, there are at most M simultaneous transmissions within each node's receive range. They use an MILP to maximize throughput and to obtain the active time fraction of each link subject to the MPR reception constraint. However, the number of transmission sets increases exponentially with the network size. Thus, the authors propose a heuristic link scheduling algorithm. In order to take advantage of MPR, they first group links with the same destination. Then, the link scheduling algorithm constructs a modified conflict graph where each vertex is a link group. The resulting conflict graph is then colored to yield transmission sets. However, finding all feasible sets is NP-hard. Therefore, the authors propose a heuristic algorithm to solve this JRS problem. The

algorithm can be separated into route selection and link scheduling. For each traffic demand, the algorithm first constructs multiple node-disjoint paths and select the one with the largest capacity and fewest hops. Then, it assigns each link to the smallest slot where the number of interfering transmissions is less than M.

Reference [63] also addresses the problem of maximizing throughput where nodes have MPR capability. Different from [62], the authors of [63] consider the physical interference model. Moreover, they showed that the scheduling sub-problem is NP-hard because it is equivalent to finding the maximum independent set (MIS) of a graph. To solve this problem in polynomial time, the key idea is to use an approximation algorithm to generate a number of sub-optimal MISs which are then used to solve a JRS LP. Specifically, they first solve an LP with the objective of maximizing throughput to find routing paths for each flow; in this LP, link capacities are given. This so called routing LP returns the amount of traffic to be routed on each link. The approximation algorithm then iteratively generates sub-optimal MISs based on the links selected by the routing LP. In each iteration, it adds a link into an MIS if all activated nodes in the MIS satisfy the MPR, half-duplex and SINR constraints. With a number of MISs in hand, the authors propose a JRS LP to maximize throughput. The LP returns the fraction of each flow that is routed on each link and the activation time of each MIS. The key constraints include flow conservation and link capacity, and the total active time of MISs that is required to meet all demands must be less than or equal to one.

Table 2.4 compares the aforementioned works according to their objective(s), transmission mode and approach. Both [27] and [62] consider the protocol interference model. Only [63] has considered the physical interference model. However, the authors of [63] assume the effects of side and back lobes can be ignored, which is a trade-off between precision and computation complexity. Another observation is that smart antenna techniques such as SDMA are similar to MTR. Consequently, smart and beam-forming antennas can be used to realize MTR WMNs.

Approaches	Objectives	JRS	Transmission Mode	Formulation	Solution
Yazdanpanah et al. [27]	Maximize throughput and minimize schedule length	Yes	BF, SDMA and SDM	LP	Column generation
Wang et al. [62]	Maximize throughput	Yes	MPR	LP	Construct node-disjoint paths and schedule links one-by-one
Crichigno et al. [63]	Maximize throughput	Yes	MPR	LP	Find sub-optimal independent sets and solve a JRS LP

Table 2.4: A comparison of works employing smart or beam-forming antennas

2.1.2.3 MIMO Antennas

In this section, all works consider nodes equipped with a MIMO antenna system [64]. Specifically, a MIMO link provides the following three characteristics [23]:

- *Array.* The capacity of a MIMO system increases linearly with the number of antennas. The total transmit/receive power can be divided amongst multiple antenna elements; alternatively, more power can be allocated to some spatial channels with better gain.

- *Diversity.* To minimize the effect of fading and to increase reliability, the same data is transmitted over multiple antenna elements. If each spatial channel has different fading, the probability of successful reception is higher when more elements are used.

- *Spatial multiplexing.* Different antenna elements can transmit different data. This means a node is able to transmit/receive multiple independent data streams simultaneously.

A key advantage of a MIMO system is that multiple independent data streams can be transmitted simultaneously. In addition, a node may use a subset of its antenna elements to null interfering transmissions. This gives rise to a key radio

constraint, whereby the total number of antennas used to transmit data and suppress interference must be less than the total number of antenna elements.

In [65], the authors formulate the MIMO JRS problem as an LP. The goal is to maximize the traffic fraction sent by each source while satisfying flow conservation, link capacity and radio constraints. A node can only transmit one data stream in each slot. Other antenna elements are used for interference cancellation. They require both ends of a link to allocate antennas to cancel interference. A key limitation of their LP is that it may produce an infeasible link schedule because the decision variables are fractional. Thus, the authors round up the activation time of each link to an integer value which can be considered as the number of allocated slots. The number of slots assigned to each link can be treated as a link's weight. The weight is then used to create a multi-graph where each edge is duplicated according to its assigned weight. Then, each link is assigned to the first feasible slot.

The authors of [66] aim to maximize network throughput via JRS and rate allocation. The main issues are interference and radio constraints. They divide the problem into two sub-problems: (i) generating transmission sets, and (ii) JRS. In (i), they first build a conflict graph and determine its chromatic index. Links with the same color can be considered as one transmission set only if there are sufficient DoFs, i.e., antenna elements, to suppress secondary interference and there is a free radio. However, with increasing network size, the number of transmission sets grows exponentially. To reduce computation time, they propose a polynomial time heuristic algorithm to generate interference-free transmission sets. The algorithm selects a link randomly and adds it into a transmission set. With the transmission sets in hand, they then use an LP to determine the routing paths and time fraction for each transmission set with the objective of maximizing the rate allocated to each flow.

In [67], the authors aim to achieve maximal end-to-end throughput in a WMN employing Digital Adaptive Array (DAA) antenna. They divide the problem into two sub-problems: (i) interference aware tree construction, and (ii) link scheduling. In (i), they assume the tree topology consists of h layers and each node belongs

to one layer. Specifically, they build a tree that minimizes the maximum node degree. To suppress secondary interference and assign antenna elements, in (ii), the authors propose a heuristic algorithm that schedules the highest loaded link first. In addition, it also assigns antenna elements to suppress secondary interference.

The authors of [68] aim to derive the minimum schedule length by considering the following issues: transmission power, antenna element weights, primary conflict and bandwidth constraint. To derive the transmission power and weights, the authors employ the distributed iterative minimum MMSE (IMMSE) beam-forming algorithm of [69]. If a set of transmissions is not feasible, the link with the lowest SINR is discarded. That is, IMMSE returns a feasible transmission set in which each link's SINR is higher than a given threshold. To avoid primary conflict and overloading links, they formulate an LP with the objective of minimizing schedule length to decide routing paths and activation time of transmission sets. They then use column generation to yield feasible transmission sets and solve the LP.

Table 2.5 summarizes the aforementioned works. Both [67] and [68] consider a tree topology rather than a general topology. Reference [68] considers the more realistic physical interference model when generating transmission sets. In this work, they adjust transmission power and antenna weights to ensure the value of SINR is higher than a threshold, meaning they do not consider using any antenna elements to suppress interference. On the other hand, references [67][65] and [66] consider the protocol model and ensure sufficient DoFs are used to suppress interference.

2.2 DoF Assignment

In this section, we focus on works that address the link scheduling and DoF assignment problems in WMNs with MIMO capability. A key feature of MIMO is that it allows a node to use a subset of its antenna elements to cancel interference to/from neighboring links in order to increase spatial reuse; the process of determining the number of antenna elements or DoFs for transmission and interference cancellation

Approaches	Objectives	JRS	Transmission Mode	Formulation	Solution
Bhatia et al. [65]	Maximize throughput	Yes	MIMO	LP	Rounding and scheduling
Liu et al. [66]	Maximize throughput	Yes	MIMO	LP	Generate transmission sets and solve a JRS LP
Xu et al. [67]	Maximize end-to-end throughput	Yes	MIMO	N/A	Construct routing tree and schedule the link with the highest load first
Cao et al. [68]	Minimize schedule length	Yes	MIMO	LP	Column generation and Lagrange decomposition

Table 2.5: A comparison of JRS works employing MIMO antenna system

is also known as *stream control* [36]. The number of data streams over a link is bounded by $MIN(K_t, K_r)$, where K_t and K_r are the DoFs of the transmitter and receiver, respectively. Instead of using all DoFs for data streams, some can be used for interference cancellation (IC), and thereby, allow multiple links to be activated simultaneously. To increase spatial reuse, interference must be nulled/suppressed at either the transmitter and/or receiver sides; i.e., a transmitter is able to null its interference to unintended receivers; a receiver can suppress the interference caused by neighboring transmitters; alternatively both can null and suppress all interference. A key constraint, however, is that the total number of elements used for data streams and IC must be less than or equal to the DoFs at each node; so called DoF constraint. Thus, a key challenge when scheduling MIMO links is how to exploit stream control to achieve specific objectives.

The following sections group works according to different link scheduling objectives: maximizing network throughput (Section 2.2.1), maximizing the minimum flow rate (Section 2.2.2) and minimizing superframe length (Section 2.2.3).

2.2.1 Maximizing Throughput

In general, the solutions in this section apply LP to generate a set of links to be activated in one slot, i.e., [70] [30] and [71]. Other works apply a contention graph to model the interference among links, i.e., [36] and [72]. With this graph in hand,

they then schedule links based on properties such as node degrees.

The authors of [70] use an LP to maximize the total flow rate sent by each source subject to half-duplex, flow conservation and DoF constraints. In this work, each node is allowed to use more than one DoF for data streams to a neighbor in an assigned slot. In addition, they consider nodes with MTR capability. The rate of a flow is decided by the number of data streams allocated to links along its path. To maximize the flow rate, the LP aims to maximize the number of data streams on each link while ensuring all interference is suppressed at the receiver side. The LP will decide which antenna of a node is used to transmit a flow and the node's transmission time slot.

The work in [30] studies the trade-off between optimizing throughput and suppressing interference. Interference is suppressed by both ends of a link. Moreover, a node can only communicate with one neighbor in a given time slot. The link rate is decided by the number of activated data streams. Thus, the problem is how to assign DoFs at each node such that the total data rate is maximized while all interference is suppressed. The authors propose an ILP with the objective of maximizing the total link data rate to determine the DoFs assignment at each node. They consider the DoF and half-duplex constraints whilst ensuring all interference among neighboring links are suppressed. This LP can be solved by any LP solver.

In [71], the authors propose a Non-Linear Program (NLP) with the objective of minimizing the total interference in a network to determine the DoFs assignment at each node. In this work, a node can only use one DoF to transmit to or receive from a neighbor in one slot and the remaining DoFs are used to null/suppress interference. The authors assume interference can be cancelled either by the transmitter or receiver. They then solve this NLP by relaxing it to an LP. In addition, the authors also proposed a distributed link coloring algorithm to generate a feasible TDMA schedule. The algorithm starts from the node with the maximal interfering neighbors. This node chooses a color for one of its links. The color of a link is feasible only if the link and all its interfering links have sufficient DoFs to suppress

each other's transmissions. After coloring all incident links, the algorithm moves to the next node and repeats the process. The links with the same color are activated in the same slot.

The authors of [72] aim to maximize spatial reuse by maximizing the number of concurrent transmissions or links. They proposed a new contention graph in which a directed edge is used to indicate a transmission interferes with another node's reception. Also, an undirected edge is used if two vertices experience primary interference. They proposed a heuristic algorithm based on the new contention graph. It first sorts all links in the contention graph according to the number of in or out-degree. A vertex/link with more in or out-edges has a higher priority. The algorithm assigns each link to the first slot where all activated nodes satisfy the half-duplex and DoF constraints. Initially, they only schedule one stream for each link. Once all links are scheduled, the algorithm then checks to see whether any links can have more than one stream. They propose a new interference suppression model called Receiver Oriented Interference Suppression (ROIS). In this model, a new stream can be added into a slot if and only if the transmitter can null its transmission to existing receivers. Based on this model, the algorithm then tries to maximize the number of streams that traverse a link.

In [25], the authors aim to maximize spatial multiplexing gain by selecting nodes and antennas with a high priority in each slot. The priority value depends on the type of data to be sent and packet delay. The authors propose a heuristic algorithm to optimize the link schedule in each slot. The algorithm starts from the stream with the highest priority and assigns it to a slot if no nodes transmit and receive simultaneously and all nodes have sufficient antennas. If a stream cannot satisfy all the constraints, it will have a higher priority in the following slots. The algorithm then considers the stream with the second highest priority. It moves to next slot after checking all streams and removing all activated streams. The superframe length is equal to the total number of time slots used to activate all streams.

Table 2.6 summarizes and compares the aforementioned works. We can see that

Approaches	Spatial Multiplexing	Interference Cancellation	MTR	Objectives	Solution
Cheng et al. [70]	Yes	Receiver Side	Yes	Maximize total flow rate	Use LP to decide DoFs assignment and link schedule
Blough et al. [30]	Yes	Transmitter or receiver side	No	Maximize total link rate	Use ILP to decide DoFs assignment
Sundaresan et al. [71]	No	Transmitter or receiver side	No	Minimize network interference	NILP and a distributed link coloring algorithm
Guo et al. [72]	Yes	Transmitter or receiver side	No	Maximize the number of concurrent transmissions	Construct a conflict graph and first schedule links with more interfering links
Chu et al. [25]	Yes	Receiver side	Yes	Maximize spatial multiplexing gain	Assign links to slots based on stream priority

Table 2.6: A comparison of works that aim to maximize throughput

[71] allows only one data stream to be used for each link in each slot. All remaining DoFs are used to null/suppress interference. Thus, the spatial multiplexing capability of MIMO is not well exploited. In addition, as shown in [29], suppressing interference at both ends of a link wastes DoFs. Moreover, as shown in [70] and [25], the MTR capability of nodes will increase network capacity further because more links are activated at the same time. The approach in [36] and [72] rely on a contention graph. The graph, however, grows rapidly with the number of links.

2.2.2 Maximizing the Minimum Flow Rate

A fundamental problem is how to exploit stream control to maximize the minimum flow rate. Existing works are based on the link layer DoF model proposed in [29]. This model, however, ensures only one end-point of a link uses its DoFs to null/suppress interference. Briefly, the model works as follows. All nodes have an order; e.g., if node A and B have order one and two, then node A is said to be *before* node B. A transmitter (receiver) only needs to null (suppress) interference to (from) receivers (transmitters) that are ordered *before* it.

In [73], the authors address the problem of maximizing the minimum flow rate by jointly optimizing routing, node ordering and DoF assignment. The authors use an MILP to decide how the DoFs at each node are assigned in order to maximize the minimum flow rate. The key constraints include half-duplex, flow conservation, link capacity, and each node must have available DoFs to null/suppress the interference caused by/to nodes before it. The authors also proposed an iterative greedy algorithm because the MILP is NP-hard. Its first step is to use a sequential fixing algorithm [74] to generate an initial feasible solution for the MILP. Then, it identifies a flow with the minimum flow rate. For each path, it also identifies the bottleneck link. The second step is to check if a data stream can be added on the bottleneck link. A data stream can be added if and only if both ends of the link have at least one remaining DoF and all interfering receivers have at least one DoF for IC. If step two fails, step three will try to reorder nodes such that step two is feasible. If all three steps fail, it means the capacity of a bottleneck link cannot be increased by adding more data streams. To increase the flow rate, in step four, the algorithm constructs a new route that avoids the bottleneck link so that more traffic can be routed between source and destination nodes.

Node ordering is also used in [75]. The authors also aim to maximize the minimum flow rate using a distributed link scheduling algorithm. The first step is to identify the bottleneck link of a flow. The bottleneck is defined as a link that needs the maximum number of DoFs for IC. The second step is to allocate DoFs to increase the rate of the selected link while ensuring all interference is cancelled according to node ordering [29]. If the second step fails, the third step will adjust the node ordering among neighboring nodes to relieve DoFs at both ends of the bottleneck such that step two is able to generate a feasible result.

2.2.3 Minimizing Superframe Length

Another problem addressed in past works is to minimize the superframe length. Specifically, the problem of allocating DoFs in a manner that yields the minimal number of time slots while satisfying the traffic demand of each link. In [76], the authors proposed a heuristic algorithm that uses a contention graph and graph coloring. They consider strong and weak contending links due to interference. Strong contending links are those that cannot be activated simultaneously because of half-duplex constraint. Weak contending links are those that can be activated if there are available DoFs. Their algorithm first constructs and colors a graph consisting only of strong links to yield p colors. They then construct p graphs, labeled as G_i. Strong links with color i are included as vertices in G_i. If these vertices interfere with one another via a weak link, then an edge is added between them in G_i. Then, they color the graph G_i, say using α colors. This means $\frac{\alpha}{K}$ slots are required to schedule all links in G_i, where K is the maximum DoFs. In each slot, links with colors $1, \ldots, K$ are included in the first slot followed by the next set of K colors and so forth. They also proposed an improvement whereby in each slot, they preferentially select the color that improves the highest marginal capacity of unsatisfied links. All interferences are suppressed at the receiver side.

The authors of [77] also address the problem of minimizing the superframe length while satisfying each link's traffic demand by determining how to assign DoFs for stream control. They proposed a heuristic algorithm based on a contention graph. The vertex with the maximum number of out-degree has the highest priority. For each vertex, the algorithm assigns it to the slot that leads to the maximum throughput improvement while ensuring in each slot, the DoF and half-duplex constraints are satisfied. Then, the algorithm schedules the next vertex with the second maximum out-degree. This step repeats until the traffic demand of all links is satisfied.

Table 2.7 summarizes the works in Section 2.2.2 and Section 2.2.3. We see that only reference [76] considers MTR. Both [76] and [77] suppress interference at the

Approaches	Spatial Multiplexing	Interference Cancellation	MTR	Objectives	Solution
Qin et al. [73]	Yes	Transmitter or receiver side	No	Maximize the minimum flow rate	An iterative greedy algorithm that jointly considers routing, scheduling and node ordering
Zeng et al. [75]	Yes	Transmitter or receiver side	No	Maximize the minimum flow rate	A distributed algorithm that jointly considers node ordering and link scheduling
Mumey et al. [76]	Yes	Receiver side	Yes	Minimize superframe length	A heuristic algorithm that uses a contention graph and graph coloring
Mumey et al. [77]	Yes	Receiver side	No	Minimize superframe length	A heuristic algorithm that uses a contention graph

Table 2.7: A comparison of works that maximize the minimum flow rate and minimize superframe length

receiver side. In addition, these two works apply a contention graph. Although references [73] and [75] generate a feasible schedule, their goal is not to generate a schedule with the minimum length.

2.3 Fair Scheduling

The works reviewed thus far have only considered end-to-end delay or network throughput. Fairness is also a key consideration when allocating resources in wireless networks. One of the most popular metrics is Max-Min Fair (MMF). A rate allocation is MMF only if the rate of a flow cannot be increased without decreasing the rate of other flows that have a smaller or equal rate [78].

In order to allocate flows subject to MMF, we must address the key challenge of determining link capacity subject to interference constraint. With the capacity in hand, we then have to assign it to flows such that the resulting rates satisfy MMF. The following sections group works according to two cases: single-hop (Section 2.3.1) and multi-hop flows (Section 2.3.2). Table 2.1 summarizes and compares all relevant works. Note that conventional MMF algorithms developed for wired networks

cannot be applied in wireless networks. This is because link capacity is not fixed and intricately tied to the link scheduler, which in turn determines the number of transmitting or interfering links [79][80].

2.3.1 Single-Hop Flow

As all works consider flows traversing one link only, the flow rate is equal to the corresponding link rate. The problem is deriving a link schedule such that the allocated link bandwidth is MMF. The authors of [79] address the MMF bandwidth allocation problem in a TDMA-based WMN where nodes form a matching in each slot. The key idea is to use tokens to control transmission rates. Each node generates tokens for all flows in a round robin fashion. The weight of a flow is the difference between the number of tokens generated and the number of packets transmitted. The weight is a measure of the gap between a flow's service and MMF rate. In each slot, the link with the heaviest weight is scheduled for service, meaning the most starved flow receives service first. After service, the corresponding token is removed. If the number of tokens for a flow exceeds a parameter W, the node will stop generating tokens for the flow.

In [81] and [82], the authors consider the same network model as [79] and propose distributed algorithms to address the MMF scheduling problem. The authors of [81] propose an asynchronous distributed algorithm with two parts: MMF rate allocation and MMF slot re-assignment. They assume the superframe length is given, meaning the network capacity is known. In the first part, each node first allocates a fair share to all its incident links; e.g., assuming all link capacity is one and nodes have three incident links. This means each node will allocate 1/3 capacity to each link. If an end node A assigns link AB with a rate a and the other end node B assigns AB with a rate b, where $a < b$, then the rate of link AB is equal to a. The remaining capacity of node B, i.e., $b - a$, will be reallocated to node B's incident links except AB. This process iterates until no link can increase its rate without decreasing the

rate of other links. In the second part, each node first assigns slots equally to all incident links. If the two end nodes of link l allocate different number of slots, then link l will be assigned the smaller allocation. Then, the node with extra slots will re-allocate its remaining slots to all its incident links except link l. This process iterates until no link can be assigned more slots.

In [82], the authors first propose a low-complexity distributed fair scheduler that constructs matchings in each slot. The key idea is to assign weights to flows that indicate their queued packets. At the start of an iteration, every node selects the incident flow with the heaviest weight and broadcasts its selection to all neighbors. If both ends of a link select the same flow, the flow will be serviced in the current slot. Otherwise, if a node finds that its neighbors selected another link, the node remains idle in the current iteration. In the next iteration, only idle nodes participate in flow selection. This process repeats until the number of iterations reaches a given threshold. Then, all selected one-hop flows, or say matchings, will be active in the current slot.

2.3.2 Multi-Hop Flow

A more realistic consideration is multi-hop or end-to-end flows. Specifically, at each intermediate node, a scheduler is required to allocate bandwidth to a flow twice because each node needs to receive and transmit packets from the flow. Consequently, if an incoming and outgoing link of a node is on the path of the same flow, they will interfere with each other; called intra-flow interference. A key issue to ensure allocated rates do not cause congestion and waste capacity. In addition, the final flow rates must satisfy MMF.

The authors of [80] consider a similar network model as [79], [81] and [82], where nodes form a matching. The difference is that they consider multi-hop flows. They proposed a centralized algorithm based on the token generation algorithm of [79]. Instead of removing tokens from the source and destination nodes after a single-flow

is serviced, the tokens are retained in order to track the token generation rate at each node along the multi-hop flow. Each node selects flows in a round-robin fashion. In slot t, node m selects flow i, and it will generate a token for i only if m has a lower token generation rate than upstream and downstream neighbors along flow i. In each time slot, the algorithm will schedule the links with the heaviest weight differential; i.e., the difference between the number of tokens at both ends of a link.

In [83], the authors address the multi-hop MMF bandwidth allocation problem in a TDMA-based multi-channel WMN where each node is equipped with multiple directional antennas with power control capability. They propose several mixed-integer programming (MIP) formulations. The key idea is to maximize each flow one at a time while keeping previously optimized flows at their optimal value. After determining the rate of each flow, they consider link scheduling as a dual problem. If a set of links improves the minimum flow rate, these links will be activated in one slot in the final schedule. In addition, they also proposed a heuristic algorithm to solve the MIP formulation for large instances. The first step is to assign channels in order to minimize interference and maximize link rate at the same time. The second step is a water-filling algorithm based on a link contention graph. They assign bandwidth equally to all flows that form a maximal clique in the contention graph. The minimum flow rate is determined by the bottleneck clique. They then remove the flows traversing the bottleneck clique from consideration. The data rate of remaining links is then recomputed, and the process is repeated until there are no more flows.

A flow contention graph is also employed by the algorithm proposed in [84]. The authors of [84] consider allocating a fair channel time to flows. Their motivation is due to two sub-problems: intra-flow contention and heterogeneous channel capacity. They first decompose a multi-hop flow into single-hop flows (sub-flows). A sub-flow contention graph is then constructed. They then identify the maximum cliques in the graph. After that, they assign equal time fraction to each multi-hop flow that has at least one sub-flow in the maximum clique. If the channel time of flow i is

larger than or equal to the channel time of other flows in a clique, then the clique is the bottleneck clique of flow i. All sub-flows belonging to the same flow must be assigned with the channel time of the bottleneck clique. They then compute the flow rate based on the channel time of each sub-flow and link capacity.

The aim of the approach in [85] is to achieve fair bandwidth allocation among competing flows in WiMAX WMNs. They assume each node only transmits to or receive from one neighbor in each time slot, and the superframe duration is given. Their solution is to determine the fair share to be allocated to each flow traversing each link subject to link capacity constraint and intra-flow interference. Consider two interfering links l_1 and l_2 with rate r_1 and r_2, respectively. Assume there are N flows that traverse these two links. Then the fair share of a sub-flow traversing link l_1 is set to $\frac{r_2}{(r_1+r_2)\times N}$, and for link l_2 it is $\frac{r_1}{(r_1+r_2)\times N}$. They then proposed a centralized algorithm that iterates through all nodes and calculates the said fair fraction for all flows that traverse each node. The resulting fair share of each link is then used to calculate the number of time slots to be assigned to each flow from the superframe used by the WiMAX WMN. Specifically, the algorithm sums up a flow's share in each traversed link and divides that by the total fair shares of all flows. The resulting fraction is then communicated to all nodes on a path, which they then use to determine the fraction of slots to assign to each passing flow.

Table 2.8 summarizes the aforementioned works. The approaches in [79] and [80] use tokens to track flow rates. Distributed heuristic algorithms are proposed in [81] and [82]. These algorithms, however, only consider single-hop flows. The algorithms proposed in [83] and [84] use a flow contention graph to derive the fair share of each flow. This, however, is computationally expensive because a flow contention graph requires checking all link pairs, where the number of link pairs increases quadratically with network size. Different from other works, the authors of [85] revised the definition of MMF to consider varying channel capacity where flows are allocated slots according to their resulting time fair fraction.

Approaches	**Single or Multiple Hop**	**Antenna Type**	**Centralized or Distributed**	**Solution**
Tassiulas et al. [79]	Single-hop	Directional or Multi-channel	Centralized	Use a token generation based algorithm to find matchings in each slot and schedule the most starved flow first
Salonidis et al. [81]	Single-hop	Directional or Multi-channel	Distributed	Each node adjusts the rate and allocated slots of all incident links
Penttinen et al. [82]	Single-hop	Directional or Multi-channel	Distributed	Slot-by-slot matching that first schedules flows with more queued packets
Sarkar et al. [80]	Multi-hop	Directional or Multi-channel	Centralized	Tracks the token generation rate of links traversed by a flow and ensures rates are fair
Pioro et al. [83]	Multi-hop	Directional	Centralized	MIP and a water-filling algorithm and uses a contention graph to identify the bottleneck flow in each iteration
Wang et al. [85]	Multi-hop	Omni-directional	Centralized	Calculate fair fraction based on interfering links and traversing flows, then assign corresponding number of slots to links
Wang et al. [84]	Multi-hop	Omni-directional	Centralized	Allocate fair channel time using flow contention graph

Table 2.8: A comparison of relevant MMF works

2.4 Random Demands

In practice, traffic demands are likely to be uncertain. Consequently, any computed link schedule or/and routing will lead to either resource waste or congestion. Moreover, uncertain demands may cause a network operator to compute and install a new routing and schedule frequently; this is likely to incur high signaling overheads, especially in large scale multi-hop WMNs.

This section focuses on works that address routing and bandwidth reservation problems with random traffic demands. It will also briefly introduce works in wired networks. The key challenge is how to use network resources efficiently and minimize reservation cost. For wireless networks, the key challenge is taking into account interference and channel capacity constraints.

2.4.1 Random Demands – Wired Networks

The main problem of interest is minimizing reserved bandwidth or cost when routing random demands. A popular goal is oblivious routing, whereby the resulting routing is suitable for all possible demands; i.e., the goal is to optimize for the worst-case performance. The main challenge, as pointed out in [86], is estimating traffic matrices (TMs). To this end, researchers have developed a computationally tractable *polyhedral model* to characterize TMs [87]. Traffic demands are described by a set of linear inequalities, and hence, the model can be used in an LP. A popular and special instance of the traffic model is the *Hose model* [88]. It provides a convenient way to specify the maximum total outgoing and incoming traffic a node can send to or receive from other nodes. As we will see in Chapter 6, this model is particularly suited for WMNs because mesh routers are responsible for aggregating traffic from clients. In other words, the total incoming and outgoing traffic of each mesh router can be succinctly described using a Hose model.

In the seminal work reported in [87], the authors first propose the polyhedral traffic model and pose the problem of finding a minimum-cost multi-path routing

over uncertain traffic demands. They propose an LP to minimize the total routing cost; i.e., the LP decides the fraction of flows passing through each link. However, the LP is semi-infinite because the number of available paths can be very large and the polyhedral set is generally an infinite set. The authors then propose an algorithm that iteratively finds routing paths and TMs. They first solve the proposed LP using shortest paths and one extreme point of the polyhedral model. They then check whether there are new paths that can be added to decrease the routing cost by testing all shortest paths between each pair of source and destination nodes. To ensure the resultant routing is robust, the authors use another LP that maximizes the traffic on each link to find new TMs that cannot be supported by the current routing. If a new path or TM is found, the algorithm will add it into the first LP and solve it again to find new routes. This process will stop when no new paths or TMs can be found.

The authors of [89] aim to find a robust routing that minimizes the total routing cost for demands described by a polyhedral model. The routing cost is the utilization of the most loaded link. The problem is to minimize link utilization subject to varying traffic demands. They consider Open Shortest Path First (OSPF) routing with the equal cost multipath (ECMP) rule. Specifically, they consider deriving the optimal routing by manipulating OSPF link weights. The authors propose a heuristic approach. In each iteration, a weight optimization step is used to find an optimized routing and link weights by taking into account the TMs in a given polyhedral model. This step uses an extension of IGP-WO [90]. Then, the algorithm uses an MIP to check whether there are any new TMs that cannot be supported by the generated routing. Such TMs are included in the next iteration. This algorithm repeats until the number of iterations reaches a given threshold.

Different from the two works in [87] and [89] that only consider routing, the work in [91] jointly considers multi-path routing and scheduling (JRS) to minimize the total reserved bandwidth when setting up a Virtual Private Network (VPN). The authors consider the Hose model and assume link capacities are bounded and given.

Approaches	JRS	Objectives	Solution
Ben-ameur et al. [87]	No	Minimize routing cost	Use an LP to decide routing and iteratively add paths and TMs
Altin et al. [89]	No	Minimize cost and ensure fairness	Find new TMs and iteratively compute the optimal routing and link weights
Erlebach et al. [91]	Yes	Minimize allocated bandwidth	Use two LPs to iteratively find new TMs and optimal routing and bandwidth allocation

Table 2.9: A comparison of works that consider random demands in wired networks

They proposed a JRS LP to determine the bandwidth that is to be reserved on each link. This also includes the fraction of traffic of each flow that is routed through each link. The objective is to minimize the bandwidth reservation cost. However, the JRS LP has infinite constraints. This motivates the authors to use a heuristic algorithm. Initially, they solve the JRS LP without traffic consideration. The solution specifies a feasible routing and bandwidth reservation. Then, in each iteration, they solve another LP that maximizes the traffic on each link to find whether there is a TM that cannot be supported by the generated routing and bandwidth reservation. If so, they add the new TM into the JRS LP. The revised LP is then used to find a new routing and bandwidth reservation. This process repeats until no new TM can be found.

Table 2.9 summarizes and compares the aforementioned works. All works use a heuristic algorithm to solve the LP because the number of constraints is infinite. We can see that only the work in [91] jointly considers routing and scheduling. However, unlike the wireless case, links in wired networks have a known and fixed capacity.

2.4.2 Random Demands – Wireless

Similar to the wired case, the problem is to decide routing paths for random traffic demands in order to minimize congestion. There are two main routing strategies: 1) predictive routing that first estimates future demands based on historical data [92], and 2) oblivious routing [93]. Another problem is JRS under dynamic traffic conditions to optimize the worst-case performance. Similar to the wired case, the

main approach is to use an LP to find optimal routing paths. All works in this section consider multi-path routing, whereby each flow uses multiple paths to route a fraction of its total traffic to the corresponding receiver.

The work in [94] and [92] balances traffic load with the aim to minimize congestion and maximize throughput. In [94], the authors consider a multi-hop wireless backbone network and assume the well-known protocol interference model. The interference set of an edge e contains all edges interfering with e and also includes e itself. The authors assume historical traffic data is available. They propose an algorithm with two main steps: traffic estimation and routing optimization. First, a method is used to reliably provide a mean value and statistic distribution of dynamic traffic demands. These two values serve as the inputs to the route optimization step. This step first considers fixed routing demand and uses an LP to maximize the minimum fraction of traffic that can be routed for each flow. The authors then extend the LP to consider uncertain traffic demands; they make use of the estimated mean value and statistic distribution. They solve the dual of the LP where the objective is to minimize the aggregated price for all interference sets. In addition, the authors also propose an approximation algorithm. For each flow, in each iteration, the algorithm finds the lowest priced path and increases its traffic load. Then, the algorithm updates path prices and repeats the process until the lowest price is higher than one. In [92], the authors extend the algorithm in [94] to MRMC WMNs where each node is able to simultaneously communicate with neighbors through multiple channels. To adapt the LP proposed in [94] to MRMC WMNs, the authors add a node radio constraint that limits the number of concurrent communications to be less than the number of radios at each node. The decision variables are the traffic fraction of each flow passing through each link and channel assignment. The authors then solve the dual of the LP with the objective of minimizing the aggregated price for all interference sets.

The objective of the work in [93] is to design an oblivious routing algorithm that minimizes network congestion and has provable worst-case performance. The net-

work congestion is defined as the maximum utilization of all interference sets, e.g., an interference set $I(e)$ containing links that interfere with link e; the set also includes the link e. The key idea is to formulate the optimal routing problem for known traffic demands and then extend the formulation to unknown traffic demands. Specifically, the authors first propose an LP with the objective of minimizing network congestion to find the optimal routing paths for fixed and known traffic demands. Then, they extend the LP to find the optimal routing that provides robust performance with uncertain traffic demands. The LP considers a large set of traffic demand vectors, where each vector can be considered as a possible combination of traffic demands. Given a routing, each traffic vector will yield a network congestion value. Thus, the LP aims to find a robust routing that minimizes the maximum network congestion over all traffic demand vectors. The LP cannot be solved directly because the number of traffic demand vectors can be infinite. To this end, the authors resort to solving the dual of the LP that aims to minimize the maximum network congestion.

The algorithm in [93] is then extended to MR-MC WMNs in [95]. The problem is to find the optimal routing that gives the minimum network congestion for all TMs in a given polyhedron. The authors use the same approach as [93]. They first formulate the problem for known traffic demands as an LP. As compared to the LP in [93], this LP has a radio constraint that limits the number of concurrent transmissions or receptions to be less than or equal to the number of radios at each node. The decision variables determine the traffic fraction of flows that is routed on each channel on each link. The authors then include a polyhedral traffic model into the LP. Unfortunately, the LP cannot be solved directly because the number of TMs in a polyhedron is infinite. Thus, the authors separate the LP into a master and slave problem. The master problem aims to minimize the maximum network congestion. The slave problem is used to find the routing that maximizes congestion for a set of traffic demands. The dual transformation of the slave problem is then included in the master problem and solved using an LP solver.

The objective of the work in [96] is to find a robust routing and channel assign-

ment that minimizes the worst-case network congestion. The authors first construct a polyhedral traffic model based on historical data. From historical data, the authors observe that traffic exhibits periodic variations. Thus, the authors propose an algorithm that characterizes traffic into a set of periodic intervals with similar profiles. The generated intervals are then used to construct a polyhedral traffic model as a model of future uncertain demands. Specifically, they use an iterative algorithm that begins with random interval sizes, and then iteratively increases or decreases the interval length. It stops when the number of iterations reaches a given threshold. With the intervals in hand, the authors then construct a polyhedral model; the inequalities describing the polyhedron serve as constraints for the LP in [95]. This LP is then used to find a robust routing that minimizes the maximum network congestion. In addition, the authors propose a heuristic channel assignment algorithm that first assigns all flows to the same channel. Then, the flow passing through the most loaded link is moved to a different channel if doing so decreases the maximum network congestion.

The authors of [97] develop a JRS algorithm to minimize the maximum congestion in WMNs under dynamic traffic conditions. The maximum congestion is defined as the maximum link utilization. The authors first find a number of maximum independent sets using a conflict graph. They assume no traffic information is available, e.g., the traffic demand between every two nodes is unknown. The authors then propose an LP to decide the fraction of demands of flows that is routed through each link and the activation time of each independent set. However, this LP cannot be solved directly because the number of TMs is infinite. Thus, the authors transform the problem into a master and slave problem. The master problem aims to minimize the maximal congestion. The slave problem aims to find the routing that maximizes link utilization for a given set of demands. A new LP can be constructed by combining the master problem and the dual form of the slave problem, which can be solved by any LP solver.

Table 2.10 summarizes and compares the aforementioned works. We can see

Approaches	**MRMC**	Interference Model	JRS	**Objectives**	**Solution**
Dai et al. [94]	No	Protocol	No	Minimize congestion and maximize throughput	Estimate future demands based on historical data and use an LP to decide routing paths
Dai et al. [92]	Yes	Protocol	No	Minimize congestion and maximize throughput	Estimate future demands based on historical data and use an LP to decide routing paths and channel assignment
Wellons et al. [93]	No	Protocol	No	Minimize the worst-case network congestion	Use an LP to determine routing paths
Wellons et al. [95]	Yes	Protocol	No	Minimize the worst-case network congestion	Use an LP to determine routing paths and channel assignment
Wellons et al. [96]	Yes	Protocol	No	Minimize the worst-case network congestion	Construct a polyhedral model based on historical data, then use an LP to decide routing paths and channel assignment
Wang et al. [97]	No	Protocol and Physical	Yes	Minimize the worst-case network congestion	Find all maximal independent sets and use an LP to decide routing paths and activation time of independent sets

Table 2.10: A comparison of works that consider random demands in WMNs

that the work in [94] [92] [93] and [95] only focuses on routing. References [94] and [92] assume historical traffic data are available, and hence, their performance is affected by the proposed traffic estimation model. The authors in [93] [95] [96] and [97] consider oblivious routing and aim to minimize the worst-case congestion. References [95] and [96] construct a polyhedral traffic model based on the knowledge of historical data.

2.5 Summary

In summary, this chapter has discussed prior TDMA-based works that consider:

1. *Joint routing and scheduling.* These approaches aim to achieve one or more objectives; examples include minimizing end-to-end delay or schedule length, maximizing network or flow throughput.

2. *DoF assignment.* In MIMO systems, link scheduling and DoF assignment go hand-in-hand. The objectives are to maximize network throughput or flow rate.

3. *Fair scheduling.* The objective of past works is to ensure single or multi-hop flows receive a fair share of the bandwidth afforded by a schedule.

4. *Random demands.* The objective of these works is to find a robust routing and bandwidth allocation with random traffic demands.

However, existing works leave the following gaps. First, there has only been a handful of works that consider end-to-end delays. Some works only focus on the MAC layer and do not propose a routing algorithm to establish paths. Moreover, they assume a tree topology is given. On the other hand, in [38], the authors propose an interference aware routing algorithm to derive the minimum schedule length. However, they only focus on the network layer. Other works jointly consider routing and scheduling, but for WMNs that use an omni-directional antenna. Hence, the performance of their approach in MTR WMNs is unknown. Second, most works that consider end-to-end fairness use a flow contention graph or a LP solver to compute the fair share of flows before deriving a link schedule. However, these approaches are computationally expensive and may yield a fair flow rate in which there is no corresponding schedule. In addition, no works have considered end-to-end fairness for TDMA-based MTR WMNs. Third, past works that consider oblivious routing over WMNs aim to establish a static routing that minimizes the worst-case congestion. However, these approaches cannot be applied over MTR WMNs because of different interference models. In addition, no one has used stochastic programming to generate a schedule that consider random demands. Fourth, most

existing works that consider the DoF assignment problem in MIMO-based WMNs do not aim to derive the minimal superframe length. For works that do, an inefficient DoF model is used for interference cancellation, meaning the derived superframe is unlikely to be optimal.

Chapter 3

Minimizing End-to-End Delays

As shown in Chapter 2, past works have either focused on maximizing network throughput in MTR WMNs [58][60][61], or have only considered minimizing end-to-end delays in WMNs where nodes have an omni-directional antenna [46][54][44][53][31]. However, no one has proposed MTR based solutions that minimize end-to-end delay through joint routing and scheduling.

To this end, this chapter considers a joint routing and link scheduling problem that addresses two fundamental issues that influence end-to-end delays: superframe length and transmission slot order. Shortening the superframe length, in terms of slots, is expected to minimize the *inter-link activation time* (IAT) whilst reordering transmission slots increases the likelihood that links on a path are activated consecutively.

This chpater presents two algorithms: JRS-Multi-DEC and JRS-BIP. Their key features include minimizing maximum link load and superframe length. They also reorder slots to minimize the average IAT such that the total end-to-end delay is minimized. Numerical results show that these algorithms are able to reduce the average end-to-end delay as compared to two algorithms, NJR and JRS-Shortest. NJR does not jointly optimize both routing and scheduling. It uses the approach by Loo et al. [32] to first generate a superframe. Demands are then routed on

their shortest path. In contrast, JRS-Shortest first decides the shortest path for each demand, and then uses Loo et al.'s approach to generate a superframe that only schedules links in the paths. In summary, this chapter makes the following contributions:

- It formulates the joint routing and scheduling problem in MTR WMNs as a nonlinear Integer Programming (NILP) problem. This formulation is novel as it is targeted at MTR WMNs, and is different from existing approaches such as[31] and [44] that focus on routers with omni-directional antennas. In addition, the MTR WMNs under consideration is also different from those that use multiple channels and multiple radios as routers operate over a single channel.

- Both JRS-Multi-DEC and JRS-BIP are the first joint routing and scheduling solutions to minimize delays for MTR WMNs. JRS-Multi-DEC uses a heuristic algorithm with a novel metric, and JRS-BIP applies a Binary Integer Program (BIP) solver to select suitable routing paths. They rely on the algorithm reported in [32] to generate a schedule. With the schedule in hand, they use a novel slot re-ordering algorithm to further reduce end-to-end delays. They can reduce the superframe length by more than 45% as compared to JRS-Shortest and more than 70% as compared to NJR. Numerical results show that both algorithms can reduce end-to-end delays by more than 50% as compared to NJR, and approximately 30% when compared against JRS-Shortest. This chapter also proves that re-ordering slots reduces end-to-end delays by at most $H(|S|-2)+1$ slots for a demand with H hops and superframe length of $|S|$.

- It shows that the theorem proposed by Dutta et al. [60] to compute the superframe length is not optimal. It also analyses the relationship between superframe length, routing paths, link weights and end-to-end delay. It shows that JRS-Multi-DEC has a computation complexity of $O(\frac{|E|\times|\mathcal{D}|}{|V|^2}\times(|E|+|V|))$, where $|E|$ is the number of links, $|V|$ is the number of nodes, $|\mathcal{D}|$ is the number

of demands. In addition, in terms of slot re-ordering, the proposed First-Hop rule, described in Section 3.3, produces lower end-to-end delays as compared to the Bucket Draining Algorithm (BDA) [32].

- It shows that routing each demand via its shortest path is the best case when there are more nodes than flows, or when $|E|$ is higher than $|V|$. Further, if a demand has multiple paths with the same length, the proposed algorithms will select the best one that leads to a lower link weight (defined in Section 3.3), which helps to reduce end-to-end delays.

3.1 Preliminaries

3.1.1 Network Model

Consider a single channel, TDMA-based MTR WMN and assume time is divided into slots. Each slot is sufficient to transmit a single packet. A superframe S is comprised of $|S|$ slots, called superframe length, whereby each link is assigned one or more slots. Let $G(V, E)$ be an arbitrary graph, where V denotes the set of nodes and E represents the set of directed links between nodes. Each router $i \in V$ is equipped with $\mathcal{A}_i \geq 1$ antennas. This chapter assumes each node is equipped with sufficient number of antennas to create a link with all neighbors. This is reasonable because systems such as Argos [98] allow a router to have up to 64 antenna elements per node. Moreover, the number of antennas can be increased by adding WARP boards[1]. Let e_{ij} or $(i, j) \in E$ denote a directional link from router i to j. This chapter defines $\mathcal{D}$ as the set of demands, indexed by d, between source $r \in V$ and destination $s \in V$, each of which has a weight R_d (in slots), where $d \in [1, |\mathcal{D}|]$. The set of available paths that can be used to route demand d is denoted as P_d. This chapter sets $|P_d| = \lceil \alpha \times \frac{|E|}{|V|^2} \rceil$. Here, α is a scalar used to control the number of alternative paths for each demand. As discussed in Section 3.4, the value of $|P_d|$ has

[1] http://warpproject.org

Notations	Description	Notations	Description
V	The set of nodes/routers/vertices	E	The set of links or edges
i	Vertex i	$\mathcal{A}_i$	The number of antennas at router i
e_{ij} or (i,j)	The directional link from i to j	f_{ij}	The total load on link (i,j)
S	The resulting superframe	$\|S\|$	Superframe length
$\mathcal{D}$	The set of end-to-end demands	d	The index of demands in $\mathcal{D}$
P_d	The set of paths used to route demand d	α	A scalar used to control the number of alternative paths for each demand
p_d^k	The k-th path in P_d	H_d^k	Number of hops in path p_d^k
$Z_{d,k}$	A binary variable set to 1 if path p_d^k is selected to route traffic	$Z_{d,k}^{ij}$	A binary variable set to 1 if path p_d^k uses link (i,j)
R_d	The number of slots required by demand d	$\mathbf{f}$	The maximal link load
t	The slot number in S	X_{ij}^t	A binary variable set to 1 if link (i,j) is activated in slot t
E_i^+	The set of node i's outgoing links	E_i^-	The set of node i's incoming links
o_z	A binary variable to ensure inter-link activation time is always positive	Ω	The set of all possible link schedules in a superframe S for two adjacent links on a path
$Y_d^{k,((i,j),(j,h))}$	The delay of the link (i,j) and (j,h) along path k of demand d	δ_d	End-to-end delay of demand d

Table 3.1: Frequently used notations

a significant impact on performance. This is because it controls the trade-off between computation time and optimality of the resulting superframe length or end-to-end delays. Each path $p_d \in P_d$ is comprised of a set of links; i.e., $p_d \subseteq E$. This chapter will refer to each path, indexed by k, in P_d as p_d^k, where $k \in [1, |P_d|]$. Let H_d^k be the number of hops in path p_d^k. Similar to [33] and [31], this chapter assumes that the demands between a pair of nodes are 'aggregate' values between the corresponding source-destination pairs.

3.1.2 Background

According to the Directed Edge Coloring (DEC) algorithm proposed in [59], if the vertices of an undirected simple graph can be colored with x colors, the correspond-

ing directed links in the graph can be colored with $\xi(x)$ colors, where $\xi(x)$ is the minimum n that satisfies $\binom{n}{\lfloor \frac{n}{2} \rfloor} \geq x$. The coloring result is a feasible MTR link schedule and the links with the same color are assigned to the same slot. Consider a complete graph of six nodes as shown in Figure 3.1 that needs $x = 6$ vertex colors. Using DEC algorithm, the minimum number of edge colors is $\xi(6) = 4$. Then, the algorithm assigns two of the four colors to each node such that six nodes have different color subsets; see the label in each node of Figure 3. In the corresponding directed graph, each directed link (i, j) is colored by selecting the color that is assigned to i (the transmitter) but not assigned to j (the receiver). For instance, link (A, B) is colored 2 which is in A's color subset but not in B's subset. Then, link (B, A) is colored 3.

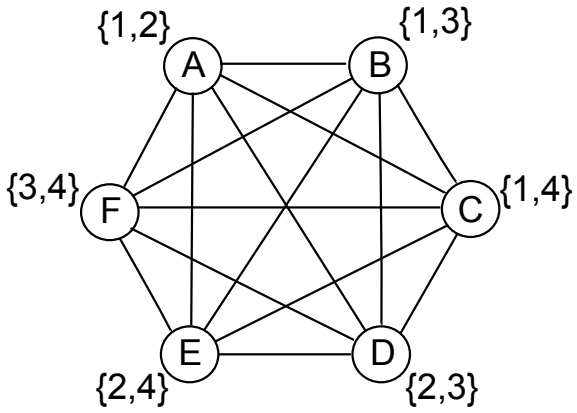


Figure 3.1: A complete graph of six nodes

In order to schedule weighted links, the authors of [60] propose Multi-DEC, which transforms the weight of a link into multiple parallel links to create a multi-graph. Then, the idea is to split the multi-graph into several simple sub-graphs and color each of them using DEC. Specifically, from [60], they proposed the following theorem:

Theorem 1. *Given a graph G, the Multi-DEC algorithm uses W_m colors when $\xi(x) = 1$, and $\lfloor \xi(x) W_m / 2 \rfloor$ colors when $\xi(x) \geq 2$, where the corresponding undirected graph of G has a chromatic index of x colors.*

Given a multi-graph, Multi-DEC separates it into several simple graphs. Each

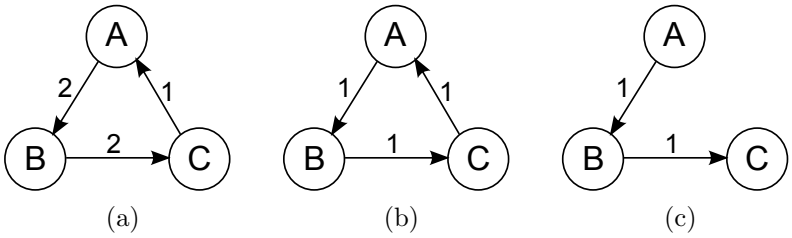


Figure 3.2: A triangle coloring example

simple graph requires $\xi(x)$ colors, where x is the chromatic index of its corresponding undirected graph. The $W_m/2$ term in expression $\lfloor \xi(x)W_m/2 \rfloor$ indicates how many simple graphs the said multi-graph can be decomposed into, and the term W_m denotes the sum of multiplicities of the most heavily loaded incoming and outgoing links of the node [60]. However, this chapter shows that the performance bound derived by Dutta et al. [60] is not optimal.

Remark 1. *The bound* $\lfloor \xi(x)W_m/2 \rfloor$ *is not optimal*

Proof. A counter example using a triangle topology shown in Figure 3.2a in which two links have a weight of two, and one link has a weight of one. Note that $x = 3$, $\xi(x) = 3$, and node B has the highest node weight at $W_m = 4$; thus Theorem 1 requires $3 * 4/2 = 6$ colors. However, the following steps shows that only five colors are needed for the link schedule of the topology. The topology can be first separated into two simple graphs, one includes three links with a weight of one, and the other one includes two links with weight of one. In the first simple graph (Figure 3.2b), Multi-DEC generates a schedule with three colors. However, in the second simple graph (Figure 3.2c), the remaining links to be activated form a bipartite graph and require only two colors. Thus, the triangle topology only requires five colors. □

Note that if a weight of two is assigned to all links in the example, Theorem 1 yields the correct number of colors. Although this algorithm is not optimal, it provides an upper bound on the minimum number of slots required by a topology. Section 3.3 shows how this bound is used in the proposed solutions.

3.2 Mathematical Model

The MTR joint routing and scheduling problem is formulated precisely using a NILP. Let $Z_{d,k}$ be a binary variable that indicates whether path k of demand d is selected to route traffic. As this chapter assumes only one path can be selected, then,

$$\sum_{p_d^k \in P_d} Z_{d,k} = 1, \quad \forall d \in \mathcal{D} \tag{3.1}$$

For example, demand $d = 1$ has three possible paths. If the first path for demand $d = 1$ is selected, then $Z_{1,1} = 1$, $Z_{1,2} = 0$, and $Z_{1,3} = 0$.

Let $Z_{d,k}^{ij}$ be a binary variable that is set to 1 if path p_d^k uses link (i, j). The next constraint ensures $Z_{d,k}^{ij}$ is one only if path k of demand d is selected,

$$Z_{d,k}^{ij} = Z_{d,k}, \quad \forall (i,j) \in p_d^k, p_d^k \in P_d \tag{3.2}$$

If link (i, j) is on the first path of demand $d = 1$, and the demand selects the first path, then $Z_{1,1}^{ij} = Z_{1,1} = 1$. The load f_{ij} on each link (i, j) is thus,

$$f_{ij} = \sum_{d \in \mathcal{D}} \sum_{p_d^k \in P_d} R_d Z_{d,k}^{ij}, \quad \forall (i,j) \in E \tag{3.3}$$

In words, the total load on edge (i, j) is determined by the paths for different demands that use said link (edge). For example, suppose link (i, j) is on the first path of demand $d = 1$. It is also on the second path of demand $d = 2$. Further, assume both demands require one slot, i.e., $R_1 = R_2 = 1$. If demand $d = 1$ selects the first path, and demand $d = 2$ selects the second path, i.e., $Z_{1,1}^{ij} = Z_{2,2}^{ij} = 1$, the total load on link (i, j) is two slots, i.e., $f_{ij} = 1 + 1 = 2$.

Let X_{ij}^t be a binary variable that indicates whether link (i, j) is activated in slot t of superframe S. The next constraint ensures that each link receives sufficient slots

to meet its total load f_{ij}.

$$\sum_{t=1}^{|S|} X_{ij}^t \geq f_{ij} \quad \forall (i,j) \in E \tag{3.4}$$

If the load on link (i,j) is two slots, and the total superframe length is four slots, then (i,j) must be activated in at least two slots. If link (i,j) is activated in slot 1 and 2, then $X_{ij}^1 = X_{ij}^2 = 1$ and $X_{ij}^3 = X_{ij}^4 = 0$. Define E_i^+ as the set of node i's outgoing links and E_i^- as the set of its incoming links. The following two constraints ensure no more than $\mathcal{A}_i$ antenna elements are activated by each node,

$$\sum_{(i,j) \in E_i^+} X_{ij}^t \leq \mathcal{A}_i, \quad \forall i \in V, \forall t \leq |S| \tag{3.5}$$

$$\sum_{(j,i) \in E_i^-} X_{ji}^t \leq \mathcal{A}_i, \quad \forall i \in V, \forall t \leq |S| \tag{3.6}$$

If node i has three antennas, it is able to concurrently transmit to or receive from up to three neighbors. To model the interference between links, the no Mix-Tx-Rx constraint of MTR WMNs needs to be considered. Consider the following example. Assume that for node i, there are $E_i^+ = \{(i,j),(i,h)\}$ and $E_i^- = \{(j,i),(h,i)\}$. This means each link pair $\{(i,j),(j,i)\}$, $\{(i,j),(h,i)\}$, $\{(i,h),(j,i)\}$, $\{(i,h),(h,i)\}$ cannot be activated in the same slot. Remark that this is distinct from the widely used protocol and interference models [18]. This is because each node i can transmit/receive to/from $\mathcal{A}_i$ distinct packets simultaneously.

To model this capability,

$$X_{ij}^t + X_{hi}^t \leq 1, \quad \forall (i,j) \in E_i^+, \forall (h,i) \in E_i^-, \quad \forall i,h,j \in V, \forall t \leq |S| \tag{3.7}$$

The ensuing set of constraints concern the delays experienced by selected paths. In particular, the activation time of links on a given path is critical. Moreover, a link may be activated multiple times within superframe S. To this end, consider the worst IAT. For example, if link e_1 is activated in slots 1 and 3, and link e_2 is activated in slot 2, and assuming a superframe length $|S| = 5$, then the worst waiting time between e_1 and e_2 is $2 - 3 + 5 = 4$, that is IAT between e_1 in slot 3 and e_2 in slot 2. The best case is clearly one slot.

Modelling the delay incurred by each pair of links is non-trivial. Consider two adjacent links on a given path. For a given superframe S, there are $2 \times \binom{|S|}{2}$ ways in which they can be activated. For example, if $|S| = 4$, then two adjacent links can be activated in one of the following 12 possible slot pairs: (1,2), (2,1), (1,3), (3,1), (1,4), (4,1), (2,3), (3,2) and so forth. Let Ω be a set of all these possible slot pairs for a superframe S. For a given path p_d^k for demand d, let $Y_d^{k,((i,j),(j,h))}$ be an integer, strictly greater than zero, that represents the delay incurred between the activation of link (i,j) and (j,h) on the said path. For each pair of said links on path p_d^k, there is,

$$Y_d^{k,((i,j),(j,h))} = \sum_{(t_a,t_b)\in\Omega} X_{ij}^{t_a} X_{jh}^{t_b} (t_b X_{jh}^{t_b} - t_a X_{ij}^{t_a} + o_z|S|), \tag{3.8}$$

where $o_z \in \{1,0\}$ models the case where the slot for link (j,h) occurs earlier than (i,j); i.e., set $o_z = 1$ when $t_a > t_b$. Specifically, observe that $Y_d^{k,((i,j),(j,h))}$ will be equal to $|S| - 1$ if link (j,h) is active one slot before link (i,j). For example, if $|S| = 4$, link (i,j) is active in slot $t_a = 3$ and link (j,h) is active in slot $t_b = 2$, then both $X_{ij}^{t_a}$ and $X_{jh}^{t_b}$ are one, $o_z = 1$ and $Y_d^{k,((i,j),(j,h))}$ is equal to $|S| - 1 = 4 - 1 = 3$. In other words, node j needs to wait for one superframe before it gets to forward the received packet from node i. Let δ_d indicate the end-to-end delay of demand d. Link (i,j) and (j,h) are two neighboring links on path k of demand d. Given

constraint (3.8), the following constraint is for path p_d^k,

$$\sum_{(i,j),(j,h)\in p_d^k} Y_d^{k,((i,j),(j,h))} \leq \delta_d + (1 - Z_{d,k})M, \quad \forall p_d^k \in P_d, \forall d \in \mathcal{D} \tag{3.9}$$

where M is a large integer, and comes into play when $Z_{d,k} = 0$. That is, if path k of demand d is not selected as part of the solution, constraint (3.9) becomes non binding.

Finally, the following objective function is to minimize the sum of delays over all demands,

$$\text{MIN} \sum_{d\in\mathcal{D}} \delta_d \tag{3.10}$$

In the aforementioned NILP, the value of $|S|$ will be specified as a parameter. That is, the NILP will be solved repeatedly, with the help of binary search, to yield a $|S|$ value with a feasible solution to the requested demands whilst minimizing end-to-end delays. Henceforth, in this chapter, it aims to consider the aforementioned issues via joint routing and link scheduling.

A superframe S can be derived by decomposing an MTR WMN into a set of bipartite graphs. Each bipartite graph contains links that are activated in one slot of S such that each edge (i, j) transmits in at least f_{ij} slots. The authors in [32] propose to generate a MAX-CUT using an approach that greedily minimizes $|S|$ and maximizes link activations in each slot. Note that MAX-CUT is a well-known NP-complete problem [99], meaning no efficient algorithms exist to compute the exact solution unless P=NP. In addition, for each router, there may be an exponential number of routes, in terms of $|V|$ and $|E|$, to a given destination. As a result, determining the combination of routes that yield the minimal end-to-end delay becomes intractable quickly with an increasing network size. Lastly, constraint (3.8) is non-linear, and each adjacent link in a path generates $2 \times \binom{|S|}{2}$ constraints.

If there is only one demand, the solution with the lowest delay is when the

demand is routed along its shortest path, and allocated time slots are consecutive. In practice, given multiple demands, selecting their shortest path may not be the best option in some cases because one or more links are likely to be occupied by several demands, e.g., link e_{dg} in Table 1.5.

3.3 Solutions

The details of JRS-Multi-DEC and JRS-BIP are now presented. In JRS-Multi-DEC, it selects routing paths that result in the least colors, as per the Multi-DEC algorithm [60], and the smallest worst case delay (WCD). The latter metric means in the worst case, each packet waits for one superframe length before it is transmitted to the next hop. In JRS-BIP, it formulates the problem as a Binary Integer Program (BIP) to minimize the sum of delays of all demands. The ultimate goal is to spread traffic demands widely to avoid overloading links, which helps reduce end-to-end delays. It is worth noting that the term 'spread' does not mean splitting one flow over several routing paths. Instead, the term refers to the assignment of paths that do not share links to demands. Both algorithms have two phases. In *Phase-1*, the problem is to select routing paths, and in *Phase-2*, the task is to schedule links and reorder the derived time slots to yield shorter end-to-end delays.

The key difference between the two proposed algorithms is in Phase-1. JRS-Multi-DEC starts by determining the shortest path for all demands in $\mathcal{D}$ and sorts them according to the number of hops in ascending order, and stores them in $\mathcal{D}'$; see *line 1-4* of Algorithm 1. Then $G'(V', E')$ is initialized to $G(V, E)$, where $V' = V$ and $E' = \emptyset$; see *line 5*. It then establishes the first demand in $\mathcal{D}'$ that has the shortest path. For each remaining demand d in $\mathcal{D}'$, it begins with the shortest path in P_d, processes each of its paths in turn, where each path is added temporarily to G' and Multi-DEC [60] is run to compute the minimum colors. Recall that this corresponds to the superframe length required to serve all established paths (or demands) and the temporarily added path; see *line 10-13* of Algorithm 1. To calculate WCD,

the superframe length is multiplied by the number of hops of a given path; see *line 14*. After testing all $|P_d|$ paths, the algorithm chooses the path whose WCD is the smallest among the $|P_d|$ paths; see *line 17-24*.

Phase-1 of JRS-BIP uses a BIP solver. It adopts Constraints (3.1), (3.2), and (3.3) in Section 3.2. The BIP, which takes an integer parameter $\mathbf{f}$, is as follows,

$$\text{MIN} \sum_{d \in \mathcal{D}} \sum_{p_d^k \in P_d} Z_{d,k} H_d^k \tag{3.11}$$

subject to:

$$f_{ij} = \sum_{d \in \mathcal{D}} \sum_{p_d^k \in P_d} R_d Z_{d,k}^{ij} \leq \mathbf{f}, \quad \forall (i,j) \in E \tag{3.12}$$

$$\sum_{p_d^k \in P_d} Z_{d,k} = 1, \quad \forall d \in \mathcal{D} \tag{3.13}$$

$$Z_{d,k}^{ij} = Z_{d,k}, \quad \forall (i,j) \in p_d^k, p_d^k \in P_d \tag{3.14}$$

The objective of the aforementioned BIP is to minimize the total path cost of all demands, where path cost corresponds to the number of hops. In addition, the load of each link must be no bigger than $\mathbf{f}$. The said BIP can be embedded within a binary search to determine a suitable $\mathbf{f}$.

Both JRS-Multi-DEC and JRS-BIP use the Algo-2 link scheduling algorithm in [32] for function LinkSchedule () in Phase-2 to derive the minimal superframe length by decomposing an MTR WMN into $|S|$ bipartite graphs. Each selected link transmits in at least f_{ij} of the $|S|$ slots; see *line 25* of Algorithm 1. *Line 25* computes the minimal superframe length, but as mentioned in Section 3.2, a sub-optimal transmission order will lead to high end-to-end delays. To address this issue, the algorithms re-order the resulting slots in decreasing number of first hops

it contains, called First-Hop rule; see *lines 26-29*. This ensures a high proportion of demands begin their transmission sooner.

Next, how JRS-Multi-DEC and JRS-BIP work is shown using the topology in Figure 3.3. Assume there are three traffic demands: from node a to g, from node b to g and from node c to g, each with $R_d = 1$. Also assume $\alpha = 25$. Then, there is $|P_d| = \lceil 25 \times \frac{10}{7^2} \rceil = 6$. To simplify exposition, only paths whose lengths are less than or equal to two hops will be considered. Thus, the following alternative paths are available: $p_1^1 = (e_{ad}, e_{dg})$, $p_2^1 = (e_{bd}, e_{dg})$, $p_2^2 = (e_{be}, e_{eg})$, $p_3^1 = (e_{cd}, e_{dg})$, $p_3^2 = (e_{cf}, e_{fg})$, each with path length 2.

Let L_d indicate the shortest path length of demand d.

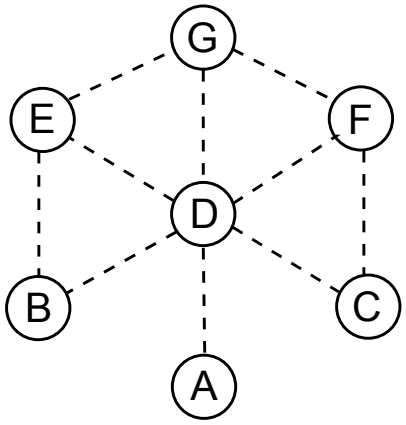


Figure 3.3: An example WMN

- *Input:* JRS-Multi-DEC takes as input the network graph $G(V, E)$, set of traffic demands $\mathcal{D}$, set of shortest paths P_d used to route demand d and corresponding number of hops H_d^k.

- *Output:* The output is $|\mathcal{D}|$ selected paths in G and a re-ordered schedule S' that contains $|S'| = |S|$ time slots; select only one path for each demand.

- *Line 1-4:* For the three demands, their respective shortest path lengths are $L_1 = L_2 = L_3 = 2$. JRS-Multi-DEC sorts the three demands in ascending order according to L_d. In this case, all three demands have the same path

Algorithm 1: JRS-Multi-DEC

input : $G(V, E)$, $\mathcal{D}$, P_d and H_d^k
output: $|\mathcal{D}|$ selected paths in G, modified schedule S'

```
   // Phase-1
 1 for d ← 1 to |D| do
 2 |   L[d] ← Distance(d)
 3 end
 4 D' ← SortAsc(L)
 5 G' ← CopyNode(G)
 6 z ← GetFirstDemand(D')
 7 Add path p_1^z into G', remove z from D'
 8 for d ∈ D' do
 9 |   for k ← 1 to |P_d| do
10 |   |   Add path p_d^k into G'
11 |   |   Color graph G' with x color
12 |   |   W_m ← MaxNodeWeight(G')
13 |   |   ncolor_k ← ⌊(ξ(x)×W_m)/2⌋
14 |   |   WCD[k] ← ncolor_k × H_d^k
15 |   |   Delete path p_d^k from G'
16 |   end
17 |   minWCD ← MIN(WCD[k])
18 |   for k ← 1 to |P_d| do
19 |   |   if WCD(k) == minWCD then
20 |   |   |   Add path p_d^k into G'
21 |   |   |   break
22 |   |   end
23 |   end
24 end
   // Phase-2
25 S ← LinkSchedule(G')
26 for s ← 1 to |S| do
27 |   nfp[s] ← NumberofFirstHop(S[s])
28 end
29 S' ← SortDesc(nfp)
```

length. Thus, in $\mathcal{D}'$, the first demand is from node a to g, the second one is b to g and third one is c to g.

- *Line 5-7:* Construct G' from G by setting $V' = V$ and $E' = \{\}$. The first demand selects path $p_1^1 = (e_{ad}, e_{dg})$. Include each link in p_1^1 in G'.

- *Line 9-16:* Demand from b to g selects $p_2^1 = (e_{bd}, e_{dg})$, two colors are needed to vertex color G'. According to the DEC algorithm [59], there is $\xi(2) = 2$. Node d has a maximum node weight of three because its maximum incoming link weight is one (e_{ad} or e_{bd}) and maximum outgoing link weight is two (e_{dg}). Thus, the number of colors as computed by Multi-DEC in *line 13* is $(2 \times 3)/2 = 3$. *Line 14* computes WCD$= 3 \times 2 = 6$ for p_2^1. If path $p_2^2 = (e_{be}, e_{eg})$ is selected, the chromatic index of the undirected graph is also two. Then, there is $\xi(2) = 2$. However, the maximum node weight is two; i.e., node e and d. Thus, the superframe length for path p_2^2 is $(2 \times 2)/2 = 2$. The WCD value of p_2^2 can be calculated as $2 \times 2 = 4$.

- *Line 17-24:* JRS-Multi-DEC then compares the WCD of p_2^1 and p_2^2, and select p_2^2 for the demand from d to g, which leads to the minimum WCD. Similarly, from Line 9-16, for the demand from c to g, the WCD of $p_3^1 = (e_{cd}, e_{dg})$ is $3 \times 2 = 6$ and the WCD of $p_3^2 = (e_{cf}, e_{fg})$ is $2 \times 2 = 4$; thus it selects p_3^2 for the demand from c to g. Upon completion of Phase-1, each demand is assigned at most one path.

- *Line 25-29:* At this phase, Algo-2 [32] is used for line 25 that produces a superframe S with links e_{eg}, e_{dg}, e_{fg} in slot-1 and links e_{be}, e_{ad}, e_{cf} in slot-2. The function in Line 27 finds that slot-2 contains the first hop of three demands, while slot-1 contains none. Given this information, Line 29 reorders the two slots in descending order according to the number of first hops, i.e., slot-2 first in the new schedule S' followed by slot-1. As the result, the end-to-end delay for the three demands using S' is 2, 2 and 2 respectively. Note that, before

reordering, i.e., slot-1 followed by slot-2 in S, for demand a to g, which travels along path $p_1^1 = (e_{ad}, e_{dg})$, its delay is $2 + 1 = 3$. For demand b to g, along path $p_2^2 = (e_{be}, e_{eg})$, its delay is $2 + 1 = 3$. Lastly, for demand c to g, which has path $p_3^2 = (e_{cf}, e_{fg})$, its delay is $2 + 1 = 3$. Thus, the first-hop reordering strategy in Phase-2 reduces the delay of each demand by one slot.

For the example above, JRS-BIP generates the same results. In constraint (3.12), the load of each link can be computed. If paths p_1^1, p_2^1 and p_3^1 are selected as mentioned in Section 3.2, the link load for e_{ad}, e_{bd} and e_{cd} is one, and e_{dg} has a load of three. If paths $p_1^1, p_2^2 = (e_{be}, e_{eg})$ and $p_3^2 = (e_{cf}, e_{fg})$ are selected, the link load for $e_{ad}, e_{be}, e_{cf}, e_{eg}, e_{dg}, e_{fg}$ is one. For each demand, only one path can be selected, as specified by constraint (3.13). Lastly, constraint (3.14) ensures $Z_{d,k}^{ij}$ is set to one only if path p_d^k is selected. In Figure 1.4, the maximum link load is one if pathsp_1^1, p_2^2 and p_3^2 are selected. Thus, the BIP will return p_1^1, p_2^2 and p_3^2 when $\mathbf{f} = 1$. In this case, the total path cost as the result of Equation (3.11) is $2 + 2 + 2 = 6$, which is minimum, and JRS-BIP produces the same schedule as JRS-Multi-DEC.

Although the aforementioned algorithms are centralized, a possible distributed solution is as follows. For the routing part, every source node can broadcast a packet to the corresponding destination node. This packet will record all nodes along the path. The destination node can then reply to the sender with all recorded routes. Note, this process is performed infrequently given that the nodes in a WMN are generally static. Each source node then sends out a test packet on each path to the destination node in order to record the highest node weight along each path. After receiving all test packets, a source node will select the route with the minimum node weight. For scheduling, the distributed link scheduling algorithm proposed in [100] can be used. The evaluation of the distributed solution is a future work.

3.4 Analysis

This section discusses several properties of JRS-Multi-DEC and JRS-BIP, and also the relationship between routing, superframe length and end-to-end delays. Lastly, it shows the time complexity of both solutions. Let *link weight*, denoted by w, be the number of demands using said link. The maximum link weight is therefore the highest number of demands using a link. Different routing paths lead to different link weights. Let G_s be the topology whereby all zero weighted links in G are removed. When multiple demands use the same link, i.e., the link's weight $w > 1$, a packet may need to wait for the link to be activated up to w times, before being served. Notice that if the link is activated once in the superframe, a packet may need to wait for up to $|S| \times w$. Based on these observations, The following discusses the relationship between routing, superframe length and end-to-end delay.

Lemma 1. *The superframe length of a schedule that routes all $|\mathcal{D}|$ demands via link disjoint paths in a topology G with chromatic index x_1 is $\xi(x_1)$.*

Proof. Since all $|\mathcal{D}|$ demands are routed via $|\mathcal{D}|$ link disjoint paths, each link is used by only one demand, and thus the maximum link weight in the resulting topology G_s is $w = 1$. Moreover, since the chromatic index of G is x_1, using [60], all links can be scheduled within $\xi(x_1)$ slots. □

Lemma 2. *The superframe length of a schedule that routes two or more demands via non-link disjoint paths in a topology G with chromatic index x_2 and the maximum link weight of w is $w \times \xi(x_2)$.*

Proof. Link weight w is proportional to the number of demands that share the link. When w demands are routed via non-link disjoint paths, there is at least one link in the disjoint paths that has weight at least two, meaning $w \in [2, |\mathcal{D}|]$. Since the resulting topology G_s needs $\xi(x_2)$ slots to activate each link at least once, the superframe length is $w \times \xi(x_2)$. □

Lemma 3. *The superframe length of the schedule that routes all $|\mathcal{D}|$ demands via link-disjoint paths (case 1) is always shorter than that of the schedule that allows non-link disjoint paths routing (case 2) if $\xi(x_1) < 2\xi(x_2)$.*

Proof. Lemma 2 shows that the minimum superframe length for case 2 is $2 \times \xi(x_2)$, i.e., $w = 2$ when only two demands share one link. Since Lemma 1 guarantees superframe length of x_1 is $\xi(x_1)$ for case 1, the schedule in case 1 is always shorter than the schedule in case 2 when $\xi(x_1) < 2\xi(x_2)$. □

The end-to-end delay is affected by the superframe length which is discussed in Section 1.2. Lemma 1, 2 and 3 show that there is a relationship between link weights, routing and superframe length. Therefore, Phase-1's objective, as carried out by JRS-Multi-DEC and JRS-BIP, is to find a proper routing path for each demand that minimizes the maximal link weight and superframe length.

Proposition 1. *The slot re-ordering algorithm in Phase-2 of JRS-Multi-DEC and JRS-BIP can potentially reduce the end-to-end delay of a demand with H hops, using schedule S, by $H(|S| - 2) + 1$ slots.*

Proof. When a packet arrives at a source node, it must wait for the slot to be assigned to its outgoing link. In the worst case, the first hop is scheduled in the $|S|$-th slot, and thus the packet incurs a delay of $|S|$ slots. Further, for this case, each outgoing link is always scheduled before the incoming link at each hop along the path, meaning the packet waits for $|S| - 1$ slots before being forwarded to each next hop. Thus, for a demand with H hops, the worst case delay is $|S| + (H-1)(|S|-1) = H(|S| - 1) + 1$ hops. Phase 2 orders slots so that the outgoing link for the first hop of each demand is scheduled earlier in the superframe. In the best case, using the order, the first hop of each demand is scheduled in the first slot, and each outgoing link is always activated in the next slot of an incoming link, i.e., there is no waiting time along each link or its IAT= 1, and thus, the optimal end-to-end delay of the demand with H hops is H slots. Consequently, the end-to-end delay of the demand

can be reduced using Phase 2 by at most $H(|S|-1)+1-H = H(|S|-2)+1$ slots. □

Proposition 2. *The computational complexity of JRS-Multi-DEC is* $O(\frac{|E|\times|\mathcal{D}|}{|V|^2} \times (|E|+|V|))$.

Proof. Referring to Algorithm 1, the time complexity of line 1-3 is $O(|\mathcal{D}|)$. Line 4 sorts the $|\mathcal{D}|$ elements in L. This step has time complexity $O(|\mathcal{D}|log|\mathcal{D}|)$. The time complexity of line 5 and 6 is $O(1)$. The computation time of line 7 depends on the number of links on the selected path. In the worst case, all links are on the selected path. Thus, the time complexity of line 1-7 is $O(|\mathcal{D}|log|\mathcal{D}|)+O(|\mathcal{D}|)+O(|E|)$. For line 11, the time complexity of greedy coloring is $O(|E|+|V|)$ [101]. The time complexity of line 12 is $O(|V|)$. The other lines, 10 to 15, are of $O(|E|)$ complexity. Thus the time complexity of line 9-16 is $O(|P_d|\times(|E|+|V|))$ or $O(\frac{|E|}{|V|^2}\times(|E|+|V|))$. For line 17, the time complexity is $O(|P_d|)$. The time complexity of line 18-23 is $O(|P_d|+|E|)$ because line 20 need $O(|E|)$. Thus, for line 9-23, the time complexity is $O(|P_d|\times(|E|+|V|))+O(|P_d|)+O(|P_d|+|E|) = O(|P_d|\times(|E|+|V|))$. For line 8-24, the time complexity is $O(|\mathcal{D}|\times|P_d|\times(|E|+|V|))$ or $O(\frac{|E|\times|\mathcal{D}|}{|V|^2}\times(|E|+|V|))$. According to [32], the time complexity of Algo-2 used in Line 25 is $O(|V|^2)$. Line 26 to 28 require $O(|S|\times|E|)$, assuming each slot contains $|E|$ links. Line 29 requires $O(|S|log|S|)$. Since $|S|$and $|E|$ in general are less than $|V|$, the time complexity of line 25-29 is $O(|V|^2)$. Therefore, the time complexity of JRS-Multi-DEC is $O(\frac{|E|\times|\mathcal{D}|}{|V|^2} \times (|E|+|V|))$. □

Proposition 3. *There are* $\alpha \times \frac{|E|\times|\mathcal{D}|}{|V|^2}$ *decision variables in Equation (3.11), and* $|E|$, $|\mathcal{D}|$, *and* $|E|$ *constraints in (3.12), (3.13) and (3.14), respectively.*

Proof. Recall that the decision variable in BIP is $Z_{d,k}$, which is set to one if the k-th path of demand d is selected. As mentioned, there are $|\mathcal{D}|$ traffic demands and $|P_d| = \alpha \times \frac{|E|}{|V|^2}$. Thus the number of decision variables is at most $|\mathcal{D}| \times |P_d|$ or $\alpha \times \frac{|E|\times|\mathcal{D}|}{|V|^2}$. According to constraint (3.13), each demand can only select one path from P_d. Thus, there are $|\mathcal{D}|$ constraint (3.13). Constraints (3.12) and (3.14)

calculate the link load of each link in G. Thus, the size of these two constraints is $|E|$. □

As a consequence of Proposition 3, JRS-BIP's running time is proportional to $|\mathcal{D}|$, $|P_d|$ or $|E|$. Specifically, if the number of demands increases to $2|\mathcal{D}|$, the number of decision variables increases to $\alpha \times \frac{|E|\times 2|\mathcal{D}|}{|V|^2} = 2 \times \alpha \times \frac{|E|\times|\mathcal{D}|}{|V|^2}$, i.e., the BIP is four times larger than before. Further, increasing the value of $|P_d|$ to $2|P_d|$ doubles the number of decision variables of constraint (3.13). Similarly, increasing $|E|$ to $2|E|$ doubles the number of constraint (3.12) and constraint (3.14).

3.5 Evaluation

The performance of JRS-Multi-DEC and JRS-BIP is evaluated using Matlab with the MatGraph toolkit [102]. Assume all nodes are stationary and randomly located on a 50×50 m^2 or 100×100 m^2 square area. The number of nodes ranges from 10 to 100. Two nodes located within each other's transmission range are neighbors. In experiments concerning transmission ranges, it varies its value from $10m$ to $50m$ with an interval of $5m$; note the transmission range used ensures nodes are separated by multiple hops within the said deployment area. Each node has a dedicated antenna/beam for each neighbor, and similar to [60] and [32], all nodes operate on the same frequency. All packet transmissions take one time slot and queues are backlogged. All demands are generated with a random source and destination node with one slot requirement. The number of demands ranges from 5 to 75. In addition, through experiments, α is set to 25. In particular, at this value, JRS-BIP and JRS-DEC have a low computation time and yet produce good solutions. First, the end-to-end delays of 15 flows are measured on a 100×100 m^2 area with 10 nodes when the transmission range of nodes is 30 meters. Then experiments are conducted to test the effect of the number of traffic demands, number of nodes, varying node degrees and transmission ranges on the superframe length, end-to-end delays and computation time. Note, only scheduling delay due to slot order and

superframe length is considered. The impact of queuing delays and retransmissions due to channel errors is deferred to a future work.

The performance of JRS-Multi-DEC and JRS-BIP are compared against JRS-Shortest and NJR. Briefly, JRS-Shortest is also a joint routing and scheduling algorithm, whereby in the routing phase, it always chooses the shortest path for each demand. NJR corresponds to a 'no joint routing' algorithm where scheduling and routing are done separately. Specifically, NJR firstly applies Algo-2 [32] to schedule all links to yield a minimal schedule with all link weights set to one. Then all packets are transmitted along the shortest path of each demand. This means a packet requires multiple superframes to traverse a route. In addition, the performance of two slot re-ordering approaches: Bucket Draining Algorithm (BDA) [32] and the First-Hop rule described in Section 3.3 are compared. BDA reorders slots based on link weights. In particular, a link with a higher weight is scheduled earlier. The First-Hop rule, described in Section 3.3, re-orders slots based on the number of links that constitute the first hop of paths. In each experiment, the following metrics are collected:

- *Superframe length.* The average total number of time slots required to satisfy all traffic demands.
- *End-to-end delay.* This records the end-to-end delay of each demand.
- *Average delay.* This records the average end-to-end delay over 50 simulation runs. In each simulation run, the average end-to-end delay over all different traffic demands is recorded.
- *Computation time.* This is the average time required by each algorithm to generate routing paths and link schedules on a computer with an Intel Core i7 with 6 GB RAM.

Each point in the forthcoming graphs is an average of 50 runs on the same topology; each run has a random set of demands. Thus, the resulting superframe

length and end-to-end delay will be different for each run. Consequently, for each plotted point, its confidence interval is also indicated.

In the first experiment, end-to-end delay of each demand is measured. There are 15 demands on a $100 \times 100\ m^2$ area with 10 nodes, each of which has a transmission range of 30 meters. For each demand, the end-to-end delay is calculated for NJR, JRS-Shortest, JRS-Multi-DEC and JRS-BIP with the following slot re-ordering algorithms: BDA and the First-Hop rule; the corresponding delays and an average of 15 demands are tabulated in Table 3.2.

Demand Number	**NJR**		**JRS-Shortest**		**JRS-Multi-DEC**		**JRS-BIP**	
	BDA	**FH**	**BDA**	**FH**	**BDA**	**FH**	**BDA**	**FH**
1	1	1	1	1	1	1	1	1
2	3	3	1	1	4	2	4	2
3	2	2	4	3	3	3	2	5
4	2	2	3	2	1	1	1	1
5	3	3	3	2	4	2	3	2
6	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1	1	1
8	7	7	9	6	6	6	6	6
9	8	8	5	5	4	2	9	4
10	8	8	5	5	5	5	5	8
11	3	3	3	2	4	2	4	2
12	13	13	11	5	5	5	9	4
13	7	7	8	6	7	4	7	5
14	3	3	5	5	5	5	4	2
15	6	6	7	7	6	6	6	6
Average	4.5333	4.5333	4.4667	3.4667	3.8000	3.0667	4.2000	3.3333

Table 3.2: End-to-end delays of 15 flows

From Table 3.2, for JRS-Shortest, JRS-Multi-DEC and JRS-BIP, using the First-Hop rule to re-order slots yields lower average end-to-end delays as compared to using BDA. For NJR, the two different slot ordering rules generate the same result in this case. JRS-Multi-DEC and JRS-BIP produce lower average delays as compared to NJR and JRS-Shortest. The use of NJR results in demand 12 incurring an end-to-end delay of 13 slots. JRS-Multi-DEC reduces the delay to five slots. JRS-BIP generates the lowest delay, incuring only four slots. We see that JRS-BIP and JRS-Multi-DEC significantly reduce the delays of demands as compared to NJR and JRS-Shortest.

The second experiment studies the effect the number of demands has on the

resulting superframe length and the average end-to-end delay. The number of demands ranges from 5 to 75 on a 50×50 m^2 topology with 50 nodes, where each node has a transmission range of 20 meters. From Figure 3.4, we see that for JRS-Shortest, JRS-Multi-DEC and JRS-BIP, the superframe length increases with traffic demands. However, for NJR, the superframe length remains fixed. Unlike other algorithms, NJR considers each link to have a weight of one, irrespective of traffic demands. This means it schedules each link once in each superframe. For the other three algorithms, the superframe length is dependent on link weights. Recall that JRS-Shortest only considers the number of hops. Consequently, it may create bottleneck links that require a higher weight. This translates to more slots per superframe when the traffic demand increases. Indeed, once there are 75 demands, the superframe length of JRS-Shortest is 0.4 slots longer than the schedule length generated by NJR. This value is about 43% and 62% longer than the one generated by JRS-Multi-DEC and JRS-BIP respectively. Comparing JRS-Multi-DEC and JRS-BIP, after the number of demands increases to 10, JRS-BIP generated superframes that are 0.04 to 0.96 slots shorter. When the number of traffic demands is 5, the superframe derived by JRS-Multi-DEC is 0.02 slot shorter than that generated by JRS-BIP. On average, the superframe length of JRS-BIP is 0.43 slot shorter than that generated by JRS-Multi-DEC

Figure 3.5a and 3.5b indicate the average end-to-end delay of NJR, JRS-Shortest, JRS-Multi-DEC and JRS-BIP after time slots are reordered using the Bucket Draining Algorithm (BDA) [32] and the First-Hop rule, respectively. Comparing these two figures, for NJR and JRS-Multi-DEC, First-Hop reduces delay by approximately 0.3 slot as compared to BDA. For JRS-Shortest and JRS-BIP, First-Hop reduces the average delay by 0.27 and 0.41 slots respectively as compared to BDA. Although BDA assigns heavily loaded links a higher priority and reallocates the corresponding slots, the transmission order of each path is not considered such that an outgoing link can be activated before an incoming link. Further, BDA aims to minimize the average delays of all possible (r, s) demands, i.e., for $|\mathcal{D}| = V(V-1)$; thus, BDA may not

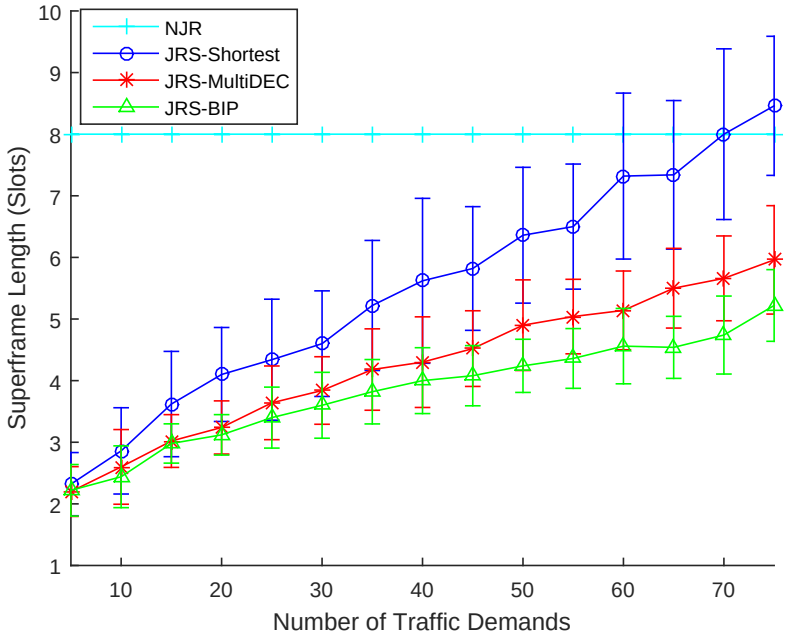
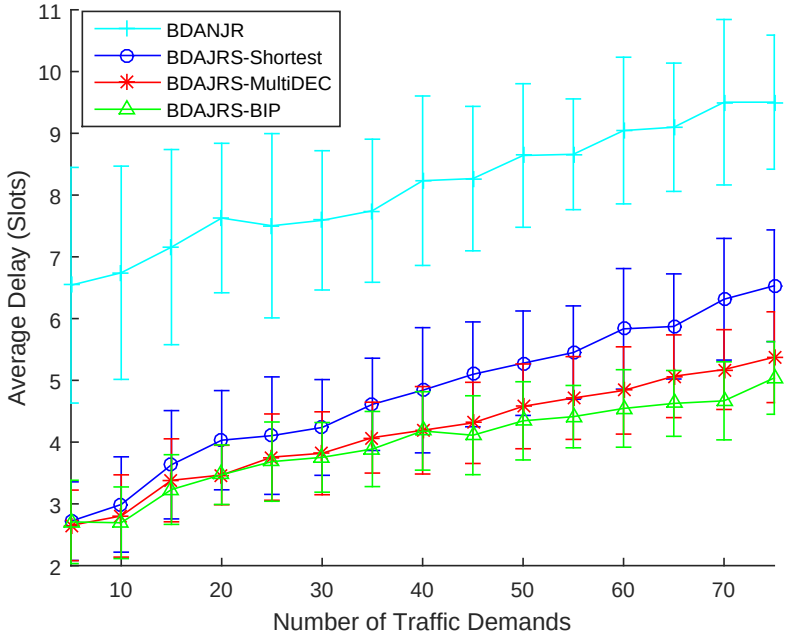


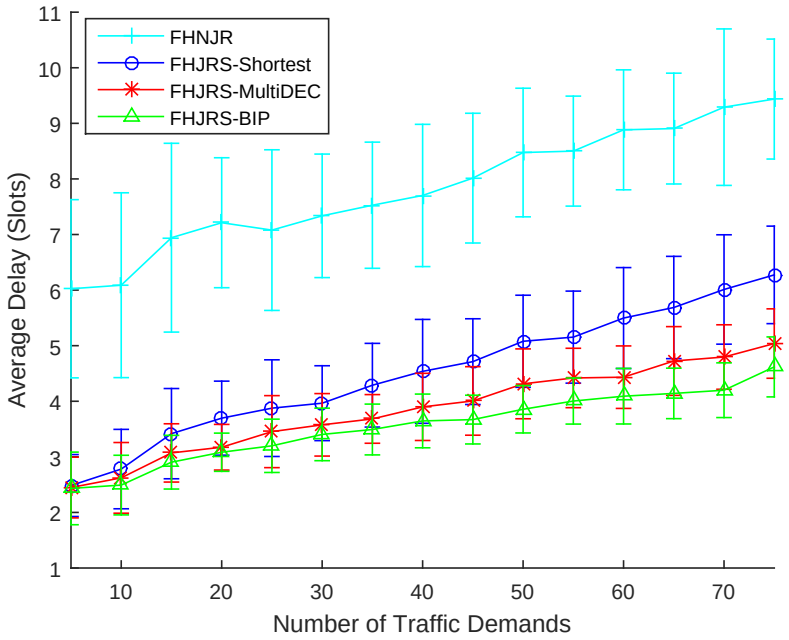
Figure 3.4: Superframe length with different number of traffic demands

minimize the end-to-end delays of some demands. NJR has the highest end-to-end delay as it only schedules each link once in a superframe. Hence, a packet has to wait for several superframes before being transmitted to its next hop if other paths also share the same outgoing link. JRS-Shortest has a much lower delay than NJR because it schedules links according to their weight. In Figure 3.5b, JRS-Multi-DEC and JRS-BIP are about 1.2 and 1.65 slots quicker than JRS-Shortest when there are 75 demands. The reason is because both algorithms result in links having a lower weight. Consequently, they have shorter superframe lengths. The average improvements are 0.65 and 0.94 slots for JRS-Multi-DEC and JRS-BIP, respectively.

The third experiment studies node numbers and their impact on the superframe length and average end-to-end delay. The number of nodes increases from 10 to 100, and they are deployed on an area of size 100×100 m^2. These nodes have a transmission range of 40 meters. The number of demands is fixed at 20. At the beginning of this experiment, it randomly regenerates a topology with 10 nodes and 20 demands. It then increases the number of nodes by randomly adding new nodes in the same topology. Recall that each point in the graph is an average



(a)



(b)

Figure 3.5: Average delay with different number of traffic demands

of 50 simulations with randomly generated demands. From Figure 3.6, we can see that for NJR, the superframe length is shorter than JRS-Shortest, JRS-Multi-DEC and JRS-BIP when the number of nodes is 10 because all links are scheduled without respect to link load. With increasing number of nodes, there will be more links, and thereby requiring proportionally longer superframe lengths. However, for JRS-Shortest, JRS-Multi-DEC and JRS-BIP, the superframe length decreases with increasing number of nodes. This is because there are more alternative paths between a source and destination pair. Moreover, the path length also reduces. Hence, link weights reduce significantly and the number of slots decreases by more than 75%. When the number of nodes increases from 10 to 100, JRS-Shortest always needs more slots than JRS-Multi-DEC and JRS-BIP. The reason is because selecting the shortest path for all demands may lead to a higher link weight and more bottleneck links which in turn require more slots. JRS-Multi-DEC requires 0.26 more slot on average as compared to JRS-BIP.

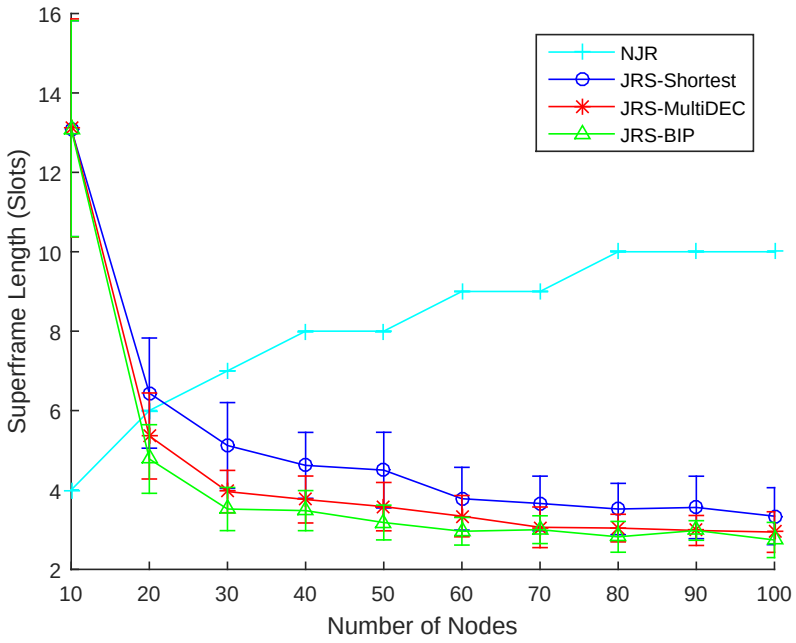


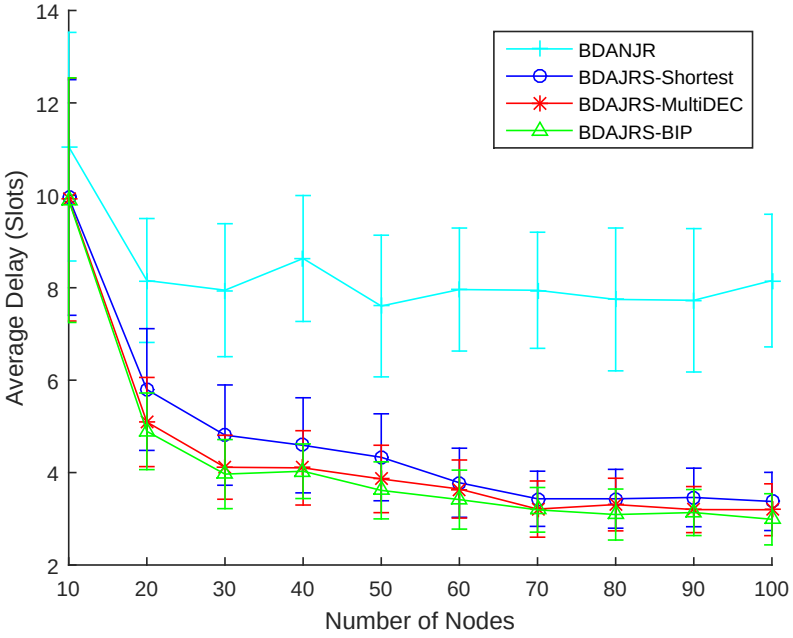
Figure 3.6: Superframe length with different number of nodes

From Figure 3.7a and 3.7b, we see that NJR has a much higher delay than the other three algorithms due to a longer superframe length. When there are

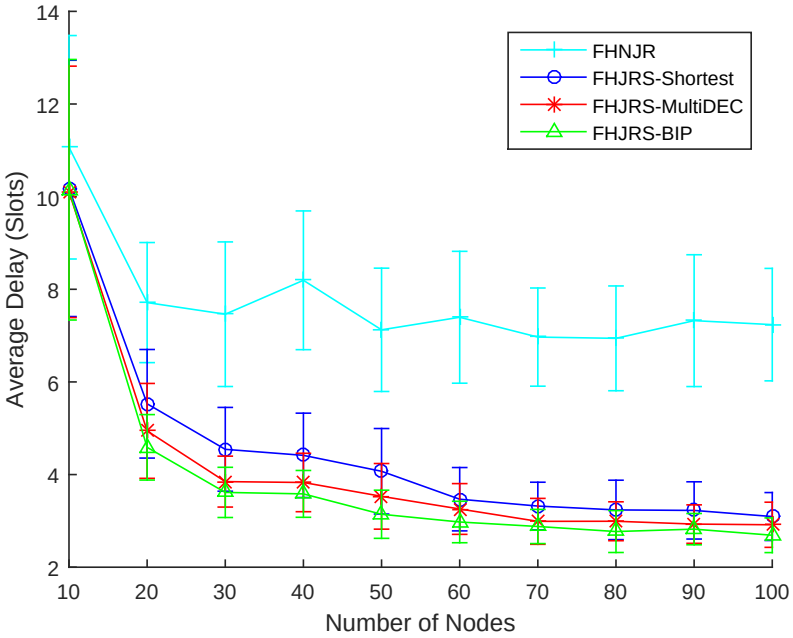
10 nodes, NJR generates shorter superframe lengths as compared to JRS-Shortest, JRS-Multi-DEC and JRS-BIP. However, the delay derived by NJR is about one slot longer than the other three algorithms because it does not consider link load. Further, when there are 10 nodes, JRS-Multi-DEC and JRS-BIP are about 0.05 slot quicker than JRS-Shortest; see Figure 3.7b. Comparing Figure 3.7a and 3.7b, First-Hop reduces the average end-to-end delay by 0.55,0.18, 0.22 and 0.30 slots for algorithms NJR, JRS-Shortest, JRS-Multi-DEC and JRS-BIP respectively. When the number of nodes increases from 20 to 100 in Figure 3.7b, JRS-Multi-DEC and JRS-BIP reduce the average delay by 0.41 to 0.65 slots as compared to JRS-Shortest. As more links are added, the probability of using the same link for different demands becomes smaller. Thus, for each demand, there are multiple shortest paths. The two proposed algorithms will select the best shortest path for each demand. In most cases, JRS-BIP is approximatelt 0.21 slot quicker than JRS-Multi-DEC.

The fourth experiment studies the effect of node degree on the superframe length and average end-to-end delay. The node degree ranges from three to seven; the number of traffic demands is fixed at 30 on a topology with 30 nodes. In this experiment, it randomly generates the topology and a set of demands at the beginning of each simulation. From Figure 3.8, when the node degree ranges from three to six, NJR has the shortest superframe length. This is because the length is dependent on the number of links. NJR has a longer superframe length, at six slots when the node degree is seven. In contrast to the other three algorithms, the superframe length of NJR only increases with higher node degrees. However, for JRS-Shortest, JRS-Multi-DEC and JRS-BIP, the superframe length decreases by more than 33% when the node degree increases because there are more alternative paths. JRS-Multi-DEC and JRS-BIP yield superframe lengths that are 0.74 and 1.19 slots shorter than JRS-Shortest respectively because they choose paths that lead to minimum link weights. Compared with JRS-Multi-DEC, JRS-BIP generates shorter superframes when nodes have more neighbors.

From both Figure 3.9a and 3.9b, we see that NJR has a lower average delay



(a)



(b)

Figure 3.7: Average delay with different number of nodes

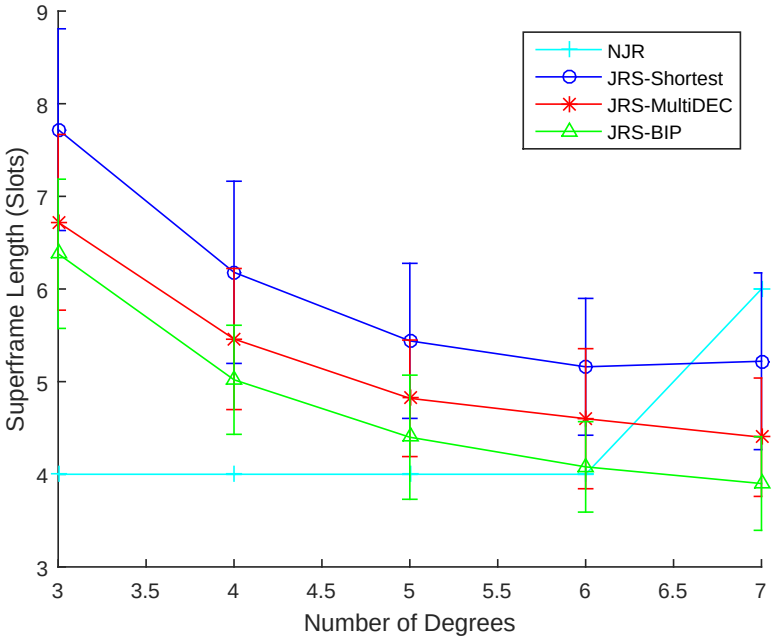


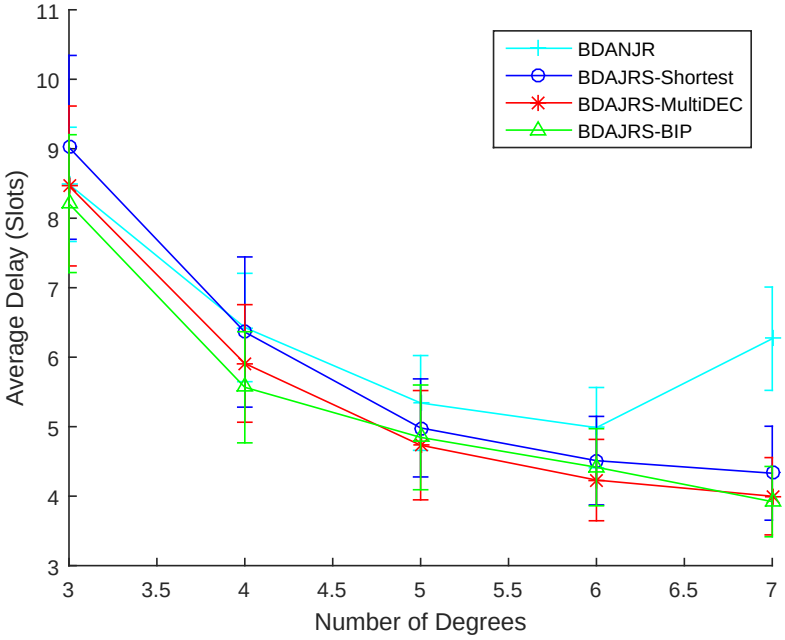
Figure 3.8: Superframe length with different node degrees

as compared to JRS-Shortest when nodes have a degree of three because it yields shorter superframe lengths. A link has to wait for the same slot in the next superframe. Thus, a shorter superframe length leads to shorter end-to-end delays. However, we see that the average delay is 6.27 and 5.63 slots when the node degree is seven in Figure 3.9a and 3.9b respectively because NJR has the longest superframe length. Comparing these two figures, we see that First-Hop is about 0.4 slots quicker than BDA. In Figure 3.9b, JRS-Multi-DEC is quicker than JRS-Shortest by about 0.45 slots. The delays experienced by demands when using JRS-BIP is shorter than JRS-Shortest by about 0.59 slots. For JRS-Shortest, JRS-Multi-DEC and JRS-BIP, the average delay reduces with increasing number of degrees. When nodes have a degree of three, there are fewer alternative paths. In addition, these alternative paths may have significantly more hops. Thus, the end-to-end delay is large. When the node degree increases, there are more alternative paths for each demand and the path length also reduces. Similar to the second experiment, using the shortest path is the best option for each demand. Compared with JRS-Shortest, JRS-Multi-DEC and JRS-BIP find the shortest paths that lead to lower link weights

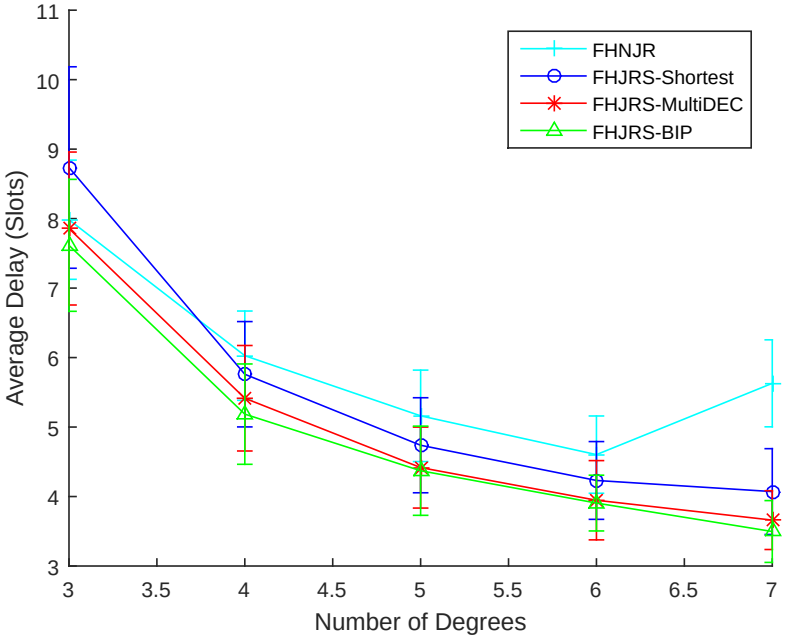
and delays.

The fifth experiment studies the effect of transmission range on the superframe length and average end-to-end delay. The topology size is 50×50 m^2; transmission range is varied from 10 to 50 meters with an interval of five meters; the number of nodes is 30 and the number of traffic demands is 30. From Figure 3.10, when the transmission range is 10 and 15 meters, NJR has the shortest superframe length because the number of links is very small. The superframe length increases or stays the same with increasing transmission range because this leads to higher node degrees. The superframe length of NJR is dependent on the number of links. With increasing number of links, for each demand, the number of alternative paths increases and path length reduces. Therefore, link weight decreases. The superframe length of JRS-Shortest, JRS-Multi-DEC and JRS-BIP reduces from 11 to 3 slots. When the transmission range is 20 meters, JRS-Multi-DEC and JRS-BIP generate superframes that are 1.38 and 1.74 slots shorter than JRS-Shortest. This is because proposed algorithms select paths that lead to lower link weights. JRS-Shortest, JRS-Multi-DEC and JRS-BIP generate the same superframe length, i.e., three slots, when the transmission range is 50 meters; note, at this range, the graph is complete. If there are a large number of links and nodes, the shortest path is the best option.

From Figure 3.11a and 3.11b, we see that NJR incurs about 0.5 slot shorter delay than JRS-Shortest, JRS-Multi-DEC and JRS-BIP because NJR has the shortest superframe length when the transmission range is 10. When the transmission range is larger than 10 meters, NJR has approximately 2.39 and 2.28 slots higher delays than the other three algorithms in Figure 3.11a and 3.11b. From these two figures, we see that for NJR and JRS-Shortest, First-Hop is 0.41 and 0.32 slots quicker than BDA. First-Hop reduces the average delay by 0.35 and 0.38 slots as compared to BDA for JRS-Multi-DEC and JRS-BIP respectively. The difference between the delays experienced by JRS-Shortest, JRS-Multi-DEC and JRS-BIP is less than 0.1 slots when the transmission range increases from 40 to 50 meters in both Figure 3.11a and 3.11b. The reason is that the transmission range has an impact on the



(a)



(b)

Figure 3.9: Average delay with different node degrees

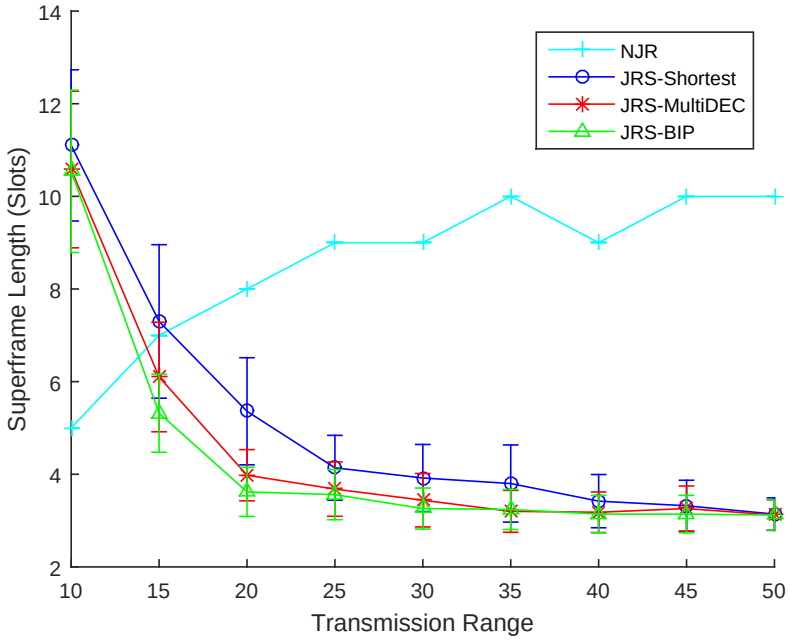
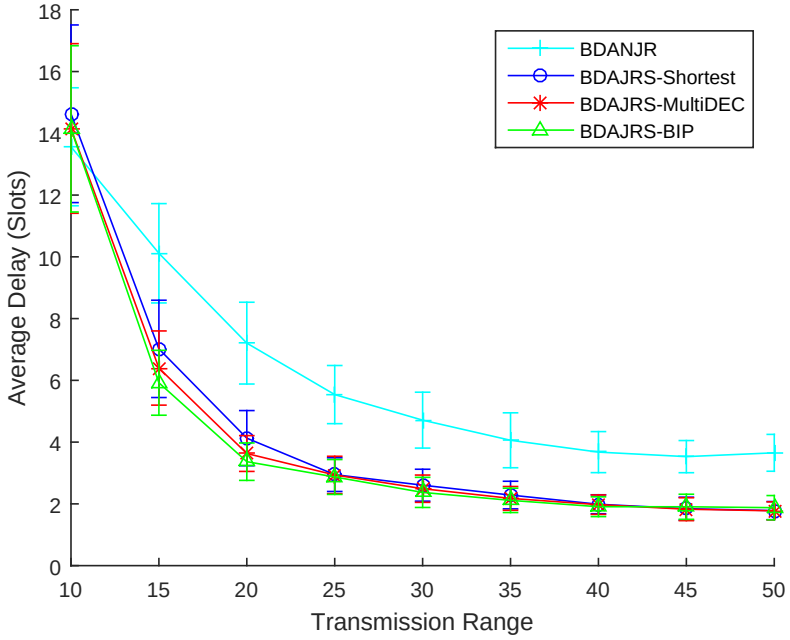


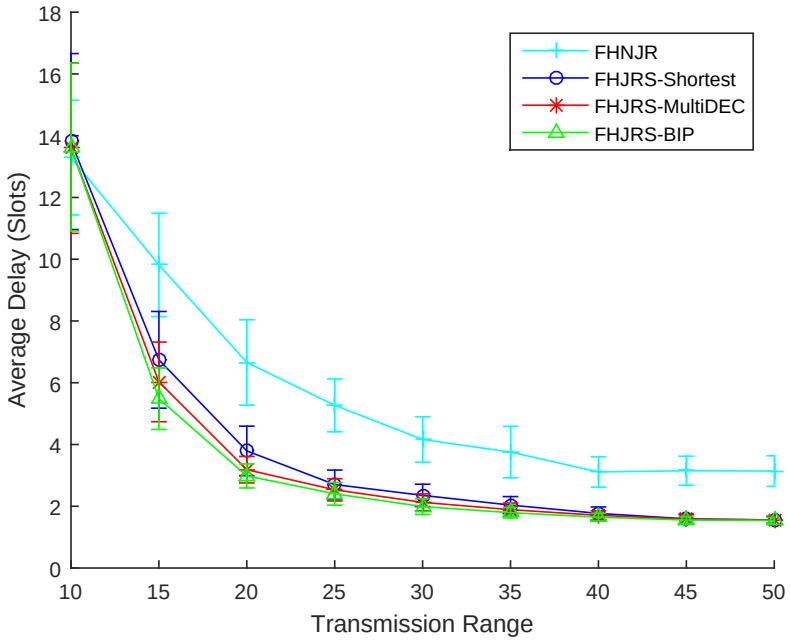
Figure 3.10: Superframe length with different transmission ranges

number of links. Longer transmission range means that there are more links in the topology. When the number of links increases, there are more alternative paths, leading to shorter path lengths. Similar to the last experiment, when the number of links is large, the best path is one of the shortest paths.

Lastly, the computation time of NJR, JRS-Shortest, JRS-Multi-DEC and JRS-BIP will be plotted on a log scale. From Figure 3.12, we see that the computation time of JRS-Multi-DEC and JRS-Shortest increases linearly with the number of demands. This agrees with the analytical results in Section 3.4. Figure 3.13, 3.14 and 3.15 indicate that the computation time of JRS-Multi-DEC does not increase proportionally with the number of nodes, node degrees and transmission range. When the number of demands is constant, the computation time of JRS-Multi-DEC depends on two factors: 1) the number of shortest paths for each demand, and 2) the number of links on selected routes; see Section 3.4. The number of shortest paths is proportional to the number of links, and inversely proportional to the square of the number of nodes; see Section 3.3. When the node degree or transmission range increases, each node will have more neighbors. Thus, there will be more links.



(a)



(b)

Figure 3.11: Average delay with different transmission ranges

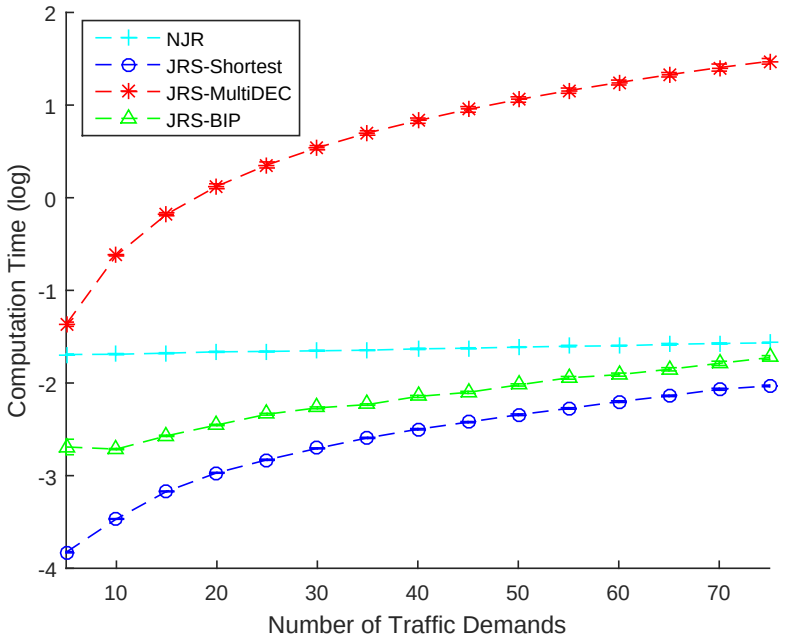


Figure 3.12: Computation time on a log scale with different number of traffic demands

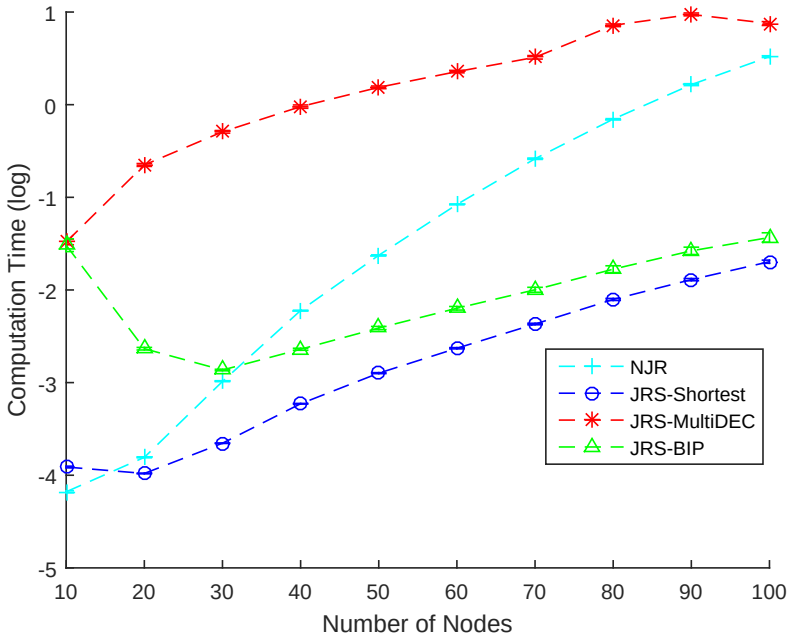


Figure 3.13: Computation time on a log scale with different number of nodes

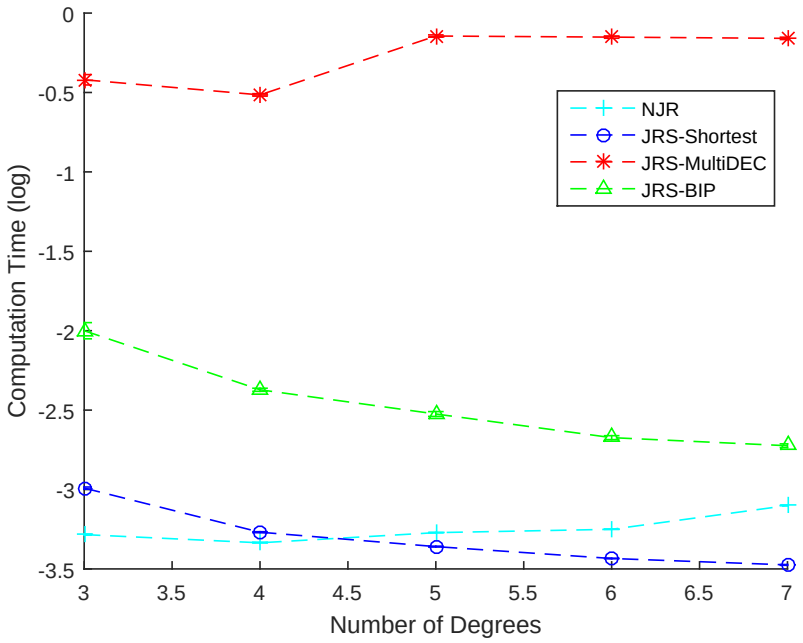


Figure 3.14: Computation time on a log scale with different node degrees

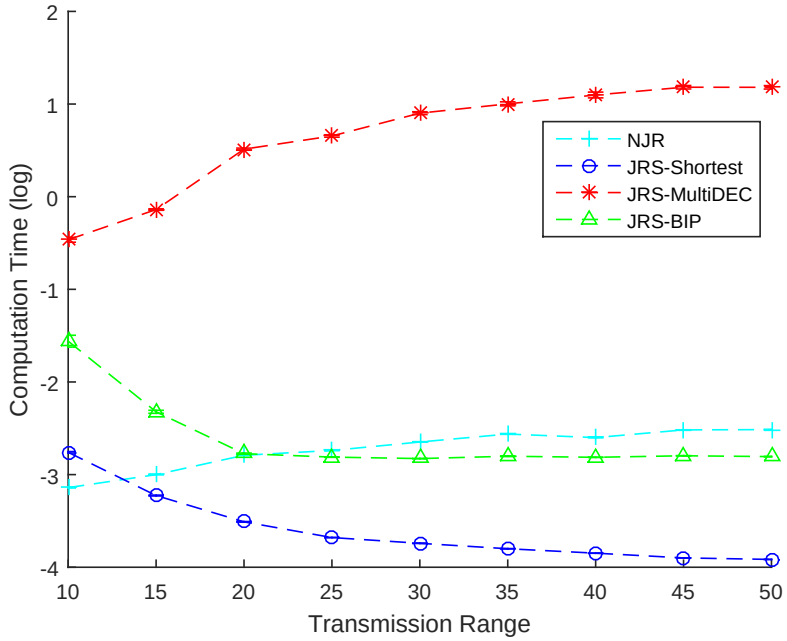


Figure 3.15: Computation time on a log scale with different transmission ranges

Consequently, as per the definition of P_d (see Section 3.3), each demand will have more shortest paths when the number of links increases such that JRS-Multi-DEC takes a longer time to find the optimal path. However, the number of hops along a shortest path will reduce when the number of links increases. Thus, JRS-Multi-DEC requires a shorter time to color the graph that includes all selected routes; see Section 3.3. JRS-Multi-DEC has a longer computation time in all four cases because it involves graph coloring. An advantage, however, is that it ensures a node with a demand that is far away from its corresponding receiver uses the shortest path. The running times of JRS-BIP is decided by the number of decision variables. A large number of decision variables causes the search space used by Matlab's Branch-and-Bound algorithm to grow exponentially with increasing number of decision variables, and results in highly variable computation time. NJR and JRS-Shortest have a shorter running time as compared to JRS-Multi-DEC and JRS-BIP because they do not use graph coloring or BIP. However, as shown in Figure 3.13, the computation time of NJR increases with the number of nodes. This is due to the longer time required to re-order slots with increasing superframe lengths.

3.6 Conclusion

This chapter has studied the problem of minimizing end-to-end delays in MTR WMNs via a joint routing and scheduling approach. Specifically, the proposed solutions consider the end-to-end delay of demands, the impact of routing, superframe length and slot ordering. The first approach employs a non-linear integer program. Unfortunately, the program is generally intractable. This is because there are an exponential number of routes. Another reason is because generating a superframe with the minimum length reduces to solving the well known MAX CUT problem in each slot. To this end, the chapter contains two novel heuristic solutions: JRS-Multi-DEC and JRS-BIP. Both solutions aim to find the optimal route combination, derive the shortest superframe and find the best slot ordering such that the average end-to-end

delay is minimized. Compared with JRS-Shortest and NJR, JRS-Multi-DEC and JRS-BIP reduce the end-to-end delay significantly when the number of demands increases. When the number of nodes or node degree increases, JRS-Multi-DEC and JRS-BIP have better performance than NJR but do not have much improvement as compared to JRS-Shortest. The proposed algorithms have better performance because they minimize the maximal node weight, which corresponds to a node's incoming and outgoing link load.

A key observation is that the proposed algorithms assume traffic demands are given. Specifically, all flows have the same demand. This assumption makes it easier to calculate end-to-end delays and ensures that flows receive the same bandwidth allocation. However, the assumption does not satisfy Max-min Fairness (MMF); a popular metric used to ensure fairness and maximize throughput. Therefore, the next chapter presents a novel link scheduling approach that derives a superframe that ensures flow rates satisfy MMF.

Chapter 4

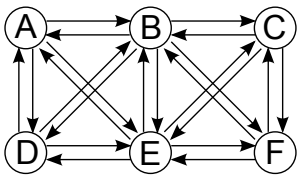
A Novel Flow Aware Link Scheduler

To date, the main *aim* of existing single channel TDMA-based MTR WMN schedulers is to derive a conflict-free schedule that minimizes the superframe length or maximizes the number of links in each slot whilst allowing each link at least one transmission opportunity or satisfies demands [21][32]. These schedulers, however, do not consider whether the fairness of flow rates. Critically, they may starve flows.

Hence, this chapter considers flow rates fairness when scheduling links. In particular, it aims to derive a schedule that yields flow rates that approximate the well-known Max-min Fair (MMF) metric. The approach taken is distinct from prior works. As discussed in Chapter 2, most existing MMF related works use a flow contention graph or an LP to compute a fair share before deriving the corresponding schedule, which may not exist. The proposed link scheduler, called Algo-Fair, however, generates a schedule directly whilst yielding an approximate MMF rate allocation. In addition, it incorporates a novel augmentation step to distribute spare capacity fairly amongst flows. This step is general and can be applied readily in other forms of wireless networks. Numerical results show that Algo-Fair is able to achieve the optimal MMF allocation in some cases, and on average, has a gap of 3.7% to the optimal solution. Moreover, Algo-Fair is able to generate a higher minimum flow rate as well as have a higher average throughput than competing approaches.

4.1 Preliminaries

Before proceeding further, three key concepts need to be defined: opportunistic links, opportunistic slots, and opportunistic flows. Consider the schedule in Figure 4.1 in which every link can be considered as a single-hop flow and is activated at least once. Observe that all transmissions in each slot adhere to the no *mix-tx-rx* constraint. Also, links AE, BD, CE and BF that have been activated in slots prior to slot 3 do not conflict with links AD, BE and CF that are currently allocated to slot 3. To maximize network capacity, these non-conflicting links can also be activated in slot 3. With this example in hand, *opportunistic links* is defined as the links that can be added into a slot without causing a conflict with other links that are already allocated to the slot; see links with an asterisk. Observe that opportunistic links have a higher number of allocated slots, meaning they have a higher capacity. An *opportunistic slot* is a slot with at least one opportunistic link; e.g., slot 3 and 4. Finally, a flow is defined as an *opportunistic flow* if *all* links on its path are opportunistic links. As an example, a two-hop flow from node A to node C via node E is an opportunistic flow because its AE and EC are opportunistic links in slot 3 and 4.



Slot 1	Slot 2	Slot 3	Slot 4
BA	AB	AD	DA
BD	DB	BE	EB
BC	CB	CF	FC
BF	FB	AE*	DB*
EA	AE	BD*	EA*
ED	DE	BF*	EC*
EC	CE	CE*	FB*
EF	FE		

Figure 4.1: A '2-boxes' topology and its TDMA MTR schedule/superframe. Opportunistic links are mark with an asterisk. A time slot containing an opportunistic link is called an opportunistic slot. A flow in which each link on its path can be activated in an opportunistic slot is called an opportunistic flow.

This chapter considers an arbitrary, single channel, TDMA-based MTR WMN $G(V, E)$, where V is the set of nodes, and E contains directed links; each link l has capacity C_l. It assumes all links have the same capacity. Further, time is divided into time slots, and each slot is sufficient to transmit a single packet. A superframe

S is defined as a grouping of slots; hence, the superframe length $|S|$ is equal to the number of slots. Note, as TDMA requires nodes to be synchronized, this chapter assumes nodes have a GPS unit.

Let $\mathcal{F}$ represent a set of flows. Each flow $f_i \in \mathcal{F}$, indexed by $i \in [1, |\mathcal{F}|]$, is routed over the shortest path, in terms of hops. With a slight abuse of notation, f_i will also be used to indicate the set of links on the path. Flows are greedy, meaning they will consume any given bandwidth. Let r_i denote the portion of bandwidth allocated to flow f_i, and r is a vector representing the rate allocated to all flows in $\mathcal{F}$; i.e., $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$. For example, $r_i = 1/3$ means all links on the route for f_i dedicate 1/3 of their capacity to the flow. Further, a link l is defined as flow f_i's *bottleneck link* if and only if link l's capacity is fully utilized by all flows crossing it and the rate of f_i is higher than or equal to the rate of other flows crossing l. Here are several definitions,

Definition 1. *A rate allocation for $\mathcal{F}$ is feasible if it does not exceed the capacity C_l of each link l.*

A schedule is *feasible* if it satisfies the following definition.

Definition 2. *A link schedule for an MTR WMN is feasible if all transmissions in each slot satisfy the no* mix-tx-rx *constraint.*

As will be elaborated in Section 4.2, the proposed approach employs Algo-2 [32], a centralized MTR link scheduler. Critically, as reported in [32], Algo-2 is able to produce a TDMA schedule that maximizes capacity with a short superframe length. To aid exposition later on, this section briefly highlights the key steps performed by Algo-2. Its goal is to generate the maximal set of links in each slot. This is achieved using a heuristic to the well-known, NP-complete, MAXCUT problem [99]. Specifically, it creates a maximal bipartite graph in each slot and derives a minimal superframe length that ensures all link demands are satisfied. In each slot k, Algo-2 creates two node sets: Set1 and Set2. The nodes in Set2 will transmit to Set1 in slot k. The two sets are constructed as follows. Initially, Algo-2 includes all nodes into

Set1 and sets Set2 to empty. It then selects the node with the highest δ value in Set1 and moves it into Set2, where the δ value of a node A is the difference between the total weight of links from A to other nodes in Set1 and the sum of link weights from nodes in Set2 to node A. It then repeats the second step until the δ value of all nodes in Set1 is less than or equal to zero. After generating one slot, the weight of activated links is reduced by one. The algorithm terminates when the weight of all links is zero. The weight of a link l corresponds to the number of slots it is assigned in the superframe generated by Algo-2. Note, Algo-2 can be replaced with any future schedulers with better performance. Notice that the proposed algorithm can also be used by other TDMA schedulers for non MTR systems. This is a subject of an immediate future work.

The '2-boxes' topology and the first scheduled slot shown in Figure 4.1 can be used to show how Algo-2 works. Assume the weight of each link is one. At the beginning, Algo-2 creates two node sets: Set1= A, B, C, D, E, F and Set2= $\emptyset$. The δ value of each node is $\{\delta_A, \delta_B, \delta_C, \delta_D, \delta_E, \delta_F\} = \{3, 5, 3, 3, 5, 3\}$. Then, Algo-2 moves node B to Set2 and updates the δ value of other nodes to $\{\delta_A, \delta_C, \delta_D, \delta_E, \delta_F\} = \{1, 1, 1, 3, 1\}$. Node D is then moved to Set2 because δ_D is the biggest. The new δ values are $\{\delta_A, \delta_C, \delta_D, \delta_F\} = \{-1, -1, -1, -1\}$. Also-2 then stops moving nodes because all δ values are negative. In the first slot, the nodes in Set2= $\{B, E\}$ will transmit to nodes in Set1= $\{A, C, D, F\}$.

The end-to-end flow fairness criterion used in this chapter is MMF [78]. Formally, it is defined as follows:

Definition 3. *A rate allocation vector $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$ is MMF for flows in $\mathcal{F}$ if, with respect to any other rate allocation vector $r' = (r'_1, r'_2, \ldots, r'_{|\mathcal{F}|})$, there exist a flow $f_i \in \mathcal{F}$ with $r_i < r'_i$, and another flow $f_j \in \mathcal{F}$ with $r_j \leq r_i$ and $r_j > r'_j$.*

Note that each flow in $\mathcal{F}$ has at least one bottleneck link, and the rate of each flow in a feasible MMF rate allocation cannot be increased without decreasing the rate of any other flows having less or equal rates [78]. In this chapter, MMF will be

used as the main fairness criterion to optimize, although other fairness metric can be used as well. A schedule is feasible and fair if it satisfies the following definition,

Definition 4. *A feasible link schedule for an MTR WMN satisfies MMF if the rate allocation for all flows in the set $\mathcal{F}$ satisfies Definition 1, 2 and 3.*

The aim of this chapter is to derive a link schedule for MTR MWNs that yields a fair rate allocation to end-to-end flows, i.e., the schedule that satisfies Definition 4. Formally, let S_n denote the set of links activated in slot $n \in [1, |S|]$, and $\mathcal{F}_l$ is the set of flows traversing link l. Thus, given an MTR WMN modeled as $G(V, E)$, a set of flows $\mathcal{F}$, the fair link scheduling problem is to find a feasible link schedule, i.e., $S = \{S_1, S_2, \ldots, S_{|S|}\}$, where (1) the rate allocation $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$ is feasible, i.e., for any link l in E, $\sum_{i \in \mathcal{F}_l} r_i \leq C_l$, and (2) the generated r is an MMF rate allocation.

Generating a rate allocation that satisfies MMF for wireless networks is more complex than wired networks because of the interference between neighboring links. Also, the resulting solution is an *approximation* because the superframe generated by Algo-2 is not optimal and hence, the capacity is sub-optimal. Nevertheless, as results in Section 4.5 show, the proposed algorithm has a better performance than other scheduling approaches.

4.2 Algo-Fair – Key Ideas

Algo-Fair first generates a *basic* superframe, using Algo-2 [32], where each flow is assigned a weight of one. This means each links of a flow f_i will be assigned one slot in the basic superframe by Algo-2, and thus Algo-Fair generates a schedule in which each link is activated in at least one slot. In the next step, Algo-Fair aims to identify opportunistic flows. A key problem in this step is that some conflicting links may have the same opportunistic slot. Consequently, they cannot be activated in the same slot. To this end, the proposed algorithm employs a superframe augmentation

step that duplicates the superframe in the prior iteration such that all conflicting links can be activated opportunistically. More importantly, previously scheduled links and thus the flow rate of non-opportunistic flows are unaffected. The algorithm terminates when the rate increase of opportunistic flows is less than or equal to a parameter called ϵ; this parameter will be elaborated in Section 4.4.

Algo-Fair creates the desired superframe iteratively, starting with the basic superframe, denoted as B_0. Figure 4.2 shows an example superframe B_n for iteration n; the *black* bars indicate the links scheduled in the superframe. Observe that superframe B_1 is constructed from B_0, and B_2 from B_1; the corresponding links in B_1 and B_2 are indicated with *white* and *dotted* bars, respectively. Let Z_n be the number of B_{n-1} copies used to construct B_n in the n-th iteration, where $Z_n \geq 1$. For example, in iteration $n = 1$, there is $Z_1 = 2$.

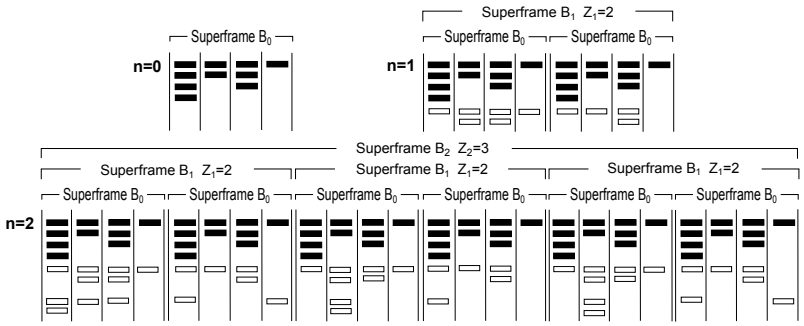


Figure 4.2: Example superframes created by proposed algorithm

Next will explain the key superframe construction or *augmentation* operation performed by Algo-Fair in iteration $n > 0$. The operation, denoted by $\bigoplus$, is defined as follows,

Definition 5. *The* $\bigoplus$ *operator on any two superframes* B_i *and* B_j*, i.e.,* $B_i \bigoplus B_j$*, appends the slots in* B_j*, and thus all links scheduled in* B_j*, into* B_i *starting at slot* $|B_i| + 1$*. Alternatively,* $B_k = B_i \bigoplus B_j$ *results in a* B_k *such that its first* $|B_i|$ *slots are exactly the same as* B_i *and its last* $|B_j|$ *slots are exactly those of* B_j*.*

Note that the $\bigoplus$ operator is non-commutative, i.e., $B_i \bigoplus B_j \neq B_j \bigoplus B_i$. For example, in Figure 4.2, superframe B_1 is generated via the operation $B_1 = B_0 \bigoplus B_0$.

The new superframe length is the total number of slots in the combined superframes. Observe that this operation preserves the flow rate of allocated flows. That is, if a link is activated in slot k of superframe B_0, the link is activated with the same period in subsequent superframes.

A very important observation is as follows. Consider an opportunistic slot k in B_0, and also two conflicting links l_1 and l_2 that can be scheduled opportunistically, but not simultaneously, in slot k. In superframe B_1, if there is only one copy of B_0, then either l_1 or l_2 can be scheduled in slot k. Now, if B_1 is augmented with another copy of B_0, then l_1 and l_2 can be scheduled in the first and second copy of B_0, respectively; these links/flows have received *additional bandwidth*. Let Δ_n denote the additional bandwidth received by all flows in iteration n. The opportunistic flows can be defined formally.

Definition 6. *A flow in iteration n is an opportunistic flow if its links can be activated in some slot(s) in superframe B_{n-1}.*

As discussed in Section 4.3, at the beginning of iteration n, Algo-Fair determines whether there are any flows where all their links can be activated opportunistically in the superframe generated in iteration $n-1$. If so, the flows are marked as opportunistic flows in iteration n. In order to keep track of opportunistic flows, it defines two sets, $F \subseteq \mathcal{F}$ and $F' \subseteq F$, which contain the opportunistic flows in superframe B_{n-1} and B_n respectively. In addition, it will use the multiset $\mathcal{L}$ to record all links used by flows in F'. Note that $\mathcal{L}$ may contain duplicated links, corresponding to links used by multiple flows in F'.

4.3 Algorithm Details

This section now delves into the details; see Algorithm 2. Algo-Fair takes as input a graph $G(V, E)$, a set of flows $\mathcal{F}$ and ϵ. Algo-Fair sets a flag m to one when it finds a flow f_i whereby all its links can be activated opportunistically in superframe

B_{n-1}. Let L be a subset of $\mathcal{L}$; i.e., $L \subseteq \mathcal{L}$, that contains all opportunistic links in slot k. Note that the links in L may conflict with each other. Hence, it uses a function called MAXCUT() to determine a maximal set of non-interfering links. Briefly, MAXCUT() divides nodes into two sets such that the total number of links between nodes in one set to nodes in the other is maximal. It outputs the links between these two sets. Let S be the final superframe returned by the algorithm.

Algo-Fair starts by assigning each flow in $\mathcal{F}$ a weight of one. Using Algo-2 [32], it generates superframe B_0 from the graph G; see *line 1-2* of Algorithm 2. Then the rate allocated to all flows in $\mathcal{F}$ is initialized to $1/|B_0|$. Recall that $|B_0|$ is the superframe length and the links of each flow is activated once in the superframe; see *line 3*. The set F is initialized to $\mathcal{F}$. Algo-Fair sets n and Δ_n to zero, and the set F' is initialized to empty; see *line 4*. In each iteration n, it includes into set F' those flows in which all their links can be opportunistic links in B_{n-1}. In addition, it includes all their links into the multiset $\mathcal{L}$; see *line 7-16*. If no flows in F can be included into F', Algo-Fair returns the superframe generated in the previous iteration; see *line 17-18, 44*. If set F' is not empty, the value of B_n is initialized to B_{n-1}, and the value of k and Z_n is set to one; see *line 20*. Let l indicate a link in set $\mathcal{L}$. In each slot k, if link l is an opportunistic link, it is included into the set L. The next step is to determine which links in L can be active simultaneously. To do this, it generates the MAX-CUT from the set L and schedules the returned links in slot k as opportunistic links, and removes the links from $\mathcal{L}$; see *line 26-27*. When all links in $\mathcal{L}$ cannot be added in the given $|B_n|$ slots, in *line 31-32*, Algo-Fair will augment B_n with B_{n-1}. This step is repeated until all links in $\mathcal{L}$ are scheduled. The number of B_{n-1} copies used in iteration n is recorded in the variable Z_n; see *line 21-35* of Algorithm 2. Recall that Δ_n is the additional bandwidth generated in each iteration. Thus the value of Δ_n is updated to $1/|B_n|$ because each flow in set F' receives one more activation in the new superframe B_n; see *line 36*. If $0 < \Delta_n < \epsilon$, the algorithm will break and return the superframe generated in the last iteration; see *line 5, 43*. At the end of each iteration n, Algo-Fair increases the allocated

bandwidth of each flow in F' by Δ_n; see *line 37-39*. Then the set F is updated to F' and F' is reset to empty; see *line 40*. The output of Algo-Fair is superframe S and vector r.

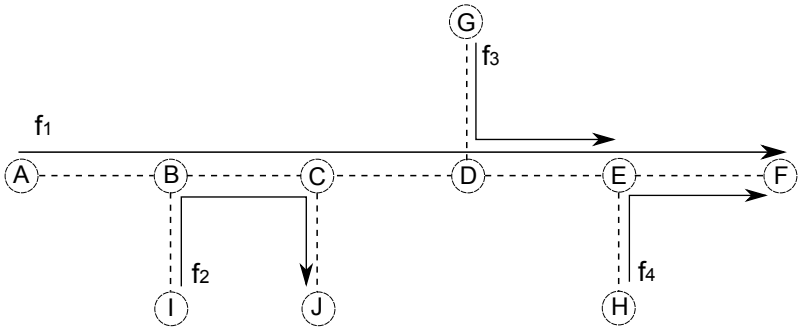


Figure 4.3: An MMF example

Slot 1	Slot 2	Slot 3	Slot 4
BC	EF	BC	EF
DE	GD	DE	
HE	CD		
	CJ		
	AB		
	IB		

Table 4.1: Basic superframe generated by Algo-2

Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7	Slot 8
BC	EF	BC	EF	BC	EF	BC	EF
DE	GD	DE	$IB*$	DE	GD	DE	$BC*$
HE	CD		$CJ*$	HE	CD		
	CJ				CJ		
	AB				AB		
	IB				IB		

Table 4.2: Final superframe generated by Algo-Fair

This section now shows how Algo-Fair computes the flow rates for the example shown in Figure 4.3.

- *Input:* It takes as input $G(V, E)$, the set of flows $\mathcal{F} = \{f_1, f_2, f_3, f_4\}$ and $\epsilon = 0.001$.
- *Output:* It outputs a superframe S and a vector r that represents the rate allocated to all flows in $\mathcal{F}$, i.e., $r = (r_1, r_2, r_3, r_4)$.
- *Line 1-4:* It assigns each flow a weight of one and generates superframe B_0 using Algo-2; the results are shown in Table 4.1. The superframe length is

$|B_0| = 4$. As flows receive one slot, each flow has a rate of $1/|B_0| = 1/4$; i.e., $r_1 = r_2 = r_3 = r_4 = 1/4$. The set F is initialized to $\mathcal{F}$, i.e., $F = \{f_1, f_2, f_3, f_4\}$. It also sets the value of n and Δ_n to zero and set F' to empty.

- *Line 7-16:* In iteration $n = 1$, link DE and EF cannot be opportunistic links in any slot in B_0, thus flow f_1, f_3 and f_4 will not be added into the set F'. On the other hand, all three links IB, BC and CJ of flow f_2 can be opportunistic links in slot 4. Thus, $F' = \{f_2\}$ and $\mathcal{L} = \{IB, BC, CJ\}$.

- *Line 17-35:* It does not return B_0 at *line 17-18* because the set F' is not empty. It then initializes the value of k and Z_n to one and $B_1 = B_0$. When $k = 4$, set L is not empty; i.e., $L = \{IB, BC, CJ\}$. To determine which links in L can be added to slot k, the function MAXCUT() is called and in this case returns two links: IB and CJ. These links are added into slot 4 and removed from set $\mathcal{L}$. Thus the only link left in $\mathcal{L}$ is BC. When the value of k is higher than $|B_1| = 4$, the algorithm augments B_1 by performing $B_1 = B_1 \bigoplus B_0$, and increases the value of Z_1 to two. Consequently, the superframe length of B_1 is 8 slots. When the value of k is 8, link BC is scheduled in slot 8 and set $\mathcal{L}$ is empty.

- *Line 36-44:* The value of Δ_n is updated to $1/|B_1| = 1/8$. Flow f_2 is the only flow in set F', thus the rate allocated to f_2, r_2 is increased by $\Delta_n = 1/8$. The total rate allocated to flow f_2 is 3/8 at the end of the first iteration. The algorithm then copies the set F' to F and set F' to empty. In the second iteration, flow f_2 is the only flow in set F. The set F' is empty after testing flow f_2 because all three links cannot be opportunistic links in B_1.

  Finally, it returns $S = B_1$ as well as $r = (r_1, r_2, r_3, r_4)$, where $r_1 = 1/4$, $r_2 = 3/8$, $r_3 = 1/4$ and $r_4 = 1/4$.

Algorithm 2: Algo-Fair

input : $G(V, E)$, $\mathcal{F}$, ϵ
output: link schedule S, rate allocation $r = (r_1, r_2, \ldots, r_{|\mathcal{F}|})$

Assign each flow $f_i \in \mathcal{F}$ a weight of one
$B_0 \leftarrow$ `Algo-2(`G`)`
$r_1 = r_2 = \cdots = r_{|\mathcal{F}|} \leftarrow 1/|B_0|$
$F \leftarrow \mathcal{F}$, $n \leftarrow 0$, $\Delta_n \leftarrow 0$, $F' \leftarrow \emptyset$
while $\Delta_n = 0$ *OR* $\Delta_n \geq \epsilon$ **do**
    $n \leftarrow n + 1$
    for $i \leftarrow 1$ **to** $|F|$ **do**
        $m \leftarrow 1$ `//` m `is a flag`
        for $l \in f_i$ **do**
            if l *cannot be an opportunistic link in any slot* **then** $m \leftarrow 0$
        end
        if $m = 1$ **then**
            Add flow f_i into F'
            Add all links of f_i into $\mathcal{L}$
        end
    end
    if $F' = \emptyset$ **then**
        break
    else
        $k \leftarrow 1$, $Z_n \leftarrow 1$, $B_n \leftarrow B_{n-1}$, $L \leftarrow \emptyset$
        while $\mathcal{L} \neq \emptyset$ **do**
            for $l \in \mathcal{L}$ **do**
                if l *can be an opportunistic link in slot* k **then** Add l into L
            end
            if $L \neq \emptyset$ **then**
                Add `MAXCUT(`L`)` in slot k
                $\mathcal{L} = \mathcal{L} \setminus$ `MAXCUT(`L`)`
            end
            $k \leftarrow k + 1$
            if $k > |B_n|$ **then**
                $B_n \leftarrow B_n \bigoplus B_{n-1}$
                $Z_n \leftarrow Z_n + 1$
            end
            $L \leftarrow \emptyset$
        end
        $\Delta_n \leftarrow 1/|B_n|$
        for $f_i \in F'$ **do**
            $r_i \leftarrow r_i + \Delta_n$
        end
        $F \leftarrow F'$, $F' \leftarrow \emptyset$
    end
end
$S \leftarrow B_{n-1}$
Return S and r

4.4 Analysis

This section now discusses several properties of Algo-Fair, and its time complexity. It proves that the superframe length in each iteration and the number of iterations are finite. Then it shows that the resulting flow rates approximate MMF. Finally, it outlines the running time complexity of Algo-Fair.

Proposition 4. *For each iteration n, Algo-Fair generates a superframe B_n with a finite length of $|B_n| = Z_n \times |B_{n-1}|$, where $Z_n \leq |\mathcal{L}| \leq |E||\mathcal{F}|$.*

Proof. Recall that $\mathcal{L}$ is a multiset containing all links belonging to opportunistic flows. As there are a finite number of links and flows, the size of $|\mathcal{L}|$ is less than or equal to $|E| \times |\mathcal{F}|$. Hence, the size of multiset $|\mathcal{L}|$ is finite. The proposed scheduler, see Algorithm 2, moves to the next iteration only if set $\mathcal{L}$ is empty. This means all links in $\mathcal{L}$ will be served in each iteration n. To see this guarantee, consider *line 21-35*. Every time Algo-Fair augments the superframe in iteration n, at least one link from $\mathcal{L}$ is scheduled in the newly copied superframe B_{n-1}. Therefore, in each iteration, the number of augmentations is finite and there are only a maximum of $|\mathcal{L}|$ such operations. □

As a result of Proposition 4, the value of Z_n is upper bounded by $|E||\mathcal{F}|$. This implies there are a maximum $|E||\mathcal{F}|$ copies of B_{n-1} in iteration n. Next, this section shows that the number of iterations is finite. It first makes the following definition.

Definition 7. *All links that are assigned a slot in the last copied B_{n-1} of superframe B_n are called 'the last links'.*

Proposition 5. *A last link l that is assigned slot k in iteration n will never be assigned the same slot k again in any later iterations.*

Proof. In iteration n, if a link l is one of the last links, then all its available opportunistic slots before the last copied B_{n-1} superframe must be occupied by conflicting links. Otherwise, link l would have been scheduled earlier; i.e., it is not a last link.

In order to schedule link l, Algo-Fair augments the superframe with a copy of B_{n-1}, meaning at least one opportunistic slot of link l is available. Otherwise, it would not have been marked an opportunistic link in iteration n. Assign link l one of these slots, say slot k. Consequently, in every copy of B_n used to construct B_{n+1}, slot k is occupied by link l. This implies in all subsequent iterations, this slot will also be occupied. This proves the statement. □

As each iteration has at least one last link, all links are guaranteed to exhaust their opportunistic slots, and thereby, at such time, they cannot be activated opportunistically. This fact leads to the next proposition.

Proposition 6. *Algo-Fair will terminate after a finite number of iterations.*

Proof. In every iteration n, there must be $x \geq 1$ last links, meaning there will be x links that lose at least one available opportunistic slot because the value of Z_n is finite, as per Proposition 4, where $1 \leq x \leq |\mathcal{L}|$. Hence, after every iteration, at least one link in $\mathcal{L}$ will lose at least one opportunistic slot. In iteration n, each link has a finite number of opportunistic slots because the number of superframes in B_{n-1} and the value of Z_n are both finite. Thus, every link will lose all its opportunistic slots after a finite number of iterations. Once there are no opportunistic slots, the set F' is empty, and Algo-Fair terminates. □

As mentioned in Section 4.2, the stopping criterion ϵ is used, where its main functionality is to reduce the computation time and to ignore negligible increase in flow rate as Z_n becomes large. Observe that n may be very large. Assume a link l in $\mathcal{L}$ has m opportunistic slots in iteration n. According to Proposition 4, the maximum number of new opportunistic slots is $|\mathcal{L}| \times (m-1)$ in iteration $n+1$. Thus, it needs at least $|\mathcal{L}| \times (m-1)$ iterations to exhaust all of link l's opportunistic slots after iteration n.

Recall that Δ_n is the additional bandwidth, due to opportunistic slots, allocated to a flow in each iteration, where Δ_n is equal to $\frac{1}{|B_n|}$. When the value of n is large,

Δ_n is very close to zero. Thus, any flow rate increase is negligible when $0 < \Delta_n < \epsilon$, and hence it is save to terminate.

To prove that the proposed scheduler approximates MMF, this section makes the following two propositions. Recall that one definition of MMF is that the rate of a flow cannot be increased without decreasing the rate of other flows with less or equal rates [78]. Also note that Algo-Fair allocates opportunistic flows the same flow rate increment in each iteration without sacrificing the rate of existing flows.

Proposition 7. *The rate allocation produced by Algo-Fair in iteration n is never less than that in iteration $n-1$.*

Proof. Assume link l is active in slot k in superframe B_{n-1} of iteration $n-1$. In the (n)-th iteration, the new superframe B_n includes one or more copies of B_{n-1}. Observe that existing allocations are preserved. This is evident in Figure 4.2. For example, the links in B_0 are not affected in each iteration. In particular, if link l is active in slot k, then its next activation will be in $k+|B_{n-1}|$, $k+|B_{n-1}|\times 2$,...,$k+|B_{n-1}|\times Z_{n-1}$. In other words, the frequency of transmission is the same. If a link is activated opportunistically, by the definition of opportunistic links, see Section 4.1, it does not increase the superframe length, and thus the rate allocation can only increase. □

To satisfy another aspect of MMF, it needs to show that every flow has at least one bottleneck link.

Proposition 8. *Every flow $f_i \notin F'$ has at least one bottleneck link.*

Proof. A link l of a flow f_i is a bottleneck if and only if link l's capacity is consumed by all flows crossing it and the rate of f_i is higher than or equal to the rate of other flows crossing l. Link l is first proved to be saturated. Recall that the number of slots allocated to a link corresponds to its capacity. Consider the case in which flow f_i was an opportunistic flow in iteration $n-1$ but is no longer one in iteration n; i.e., it is not in the set F'. By the definition of F', flow f_i has at least one link that

does not have an opportunistic slot in B_{n-1}; assume this to be link l. This means no more slots can be allocated to link l to increase its capacity; i.e., it is saturated. Secondly, the rate of f_i is proved to be higher than or equal to the rate of other flows crossing l. Consider the case where flow f_i is not the only flow crossing link l. Let there be z flows that were in F' in iteration $n-1$ but are no longer in F' in iteration n. Referring to *line 37-39* in Algorithm 2, all flows in F' will receive the same additional bandwidth in iteration 1 to $n-1$. This means all z flows and f_i will be allocated the same bandwidth up to iteration $n-1$. Then the maximum rate allocated to these z flows is $1/|B_0| + 1/|B_1| + \cdots + 1/|B_{n-1}|$. Thus, the rate of f_i is higher than or equal to the rate of other flows crossing l, meaning l is a bottleneck link. □

Finally, following time complexity result is shown below.

Proposition 9. *Algo-Fair has a time complexity of* $O(|E| \times |\mathcal{F}| \times |V|^2)$.

Proof. Consider Algorithm 2. The time complexity of line 1 and 3 is $O(|\mathcal{F}|)$. Line 2 needs to run Algo-2, which has a time complexity is $O(|V|^2)$. Thus, the time complexity of line 1-4 is $O(|\mathcal{F}|) + O(|V|^2) + O(|\mathcal{F}|) + O(1) + O(1) + O(1) + O(1)$. Assuming $|\mathcal{F}| < |V|^2$, the complexity is $O(|V|^2)$. Lines 9-11 need to check if each link of flow f_i is an opportunistic link in all slots of B_{n-1}. Thus, the complexity of lines 9-11 is $O(|E| \times |B_{n-1}|)$. The time complexity of lines 13-14 is $O(|E|)$. The iteration from lines 7-16 needs to check all flows in F. Thus, the time complexity is $O(|F| \times |E| \times |B_{n-1}|)$, where $|B_{n-1}| = |B_0| \times Z_1 \times Z_2 \times \cdots \times Z_{n-1}$. The time complexity of lines 17-20 is $O(1)$. Lines 22-24 need to check each link in $|\mathcal{L}|$, where $|\mathcal{L}| \leq |E| \times |\mathcal{F}|$. The complexity of lines 25-28 is $O(|V|^2)$ because the MAXCUT() function uses Algo-2. Lines 29-34 has complexity $O(1)$. The total time complexity of lines 22-34 is $O(|E| \times |\mathcal{F}|) + O(|V|^2) + O(1)$. If the value of $|\mathcal{F}|$ is less than $\frac{|V|^2}{|E|}$, then it becomes $O(|V|^2)$. The maximum number of iterations from lines 21 to 35 is $|\mathcal{L}|$. Thus, the time complexity of line 21-35 is $O(|E| \times |\mathcal{F}| \times |V|^2)$. The time complexity of line 36-40 is $|F'|$. Thus the total time complexity of Algorithm 2 is

$O(|F| \times |E| \times |B_{n-1}|) + O(1) + O(|E| \times |\mathcal{F}| \times |V|^2) + O(|F'|)$. If $|B_{n-1}| < |V|^2$, then the time complexity of the proposed MMF scheduling algorithm is $O(|E| \times |\mathcal{F}| \times |V|^2)$. □

4.5 Evaluation

The experiments are conducted using Matlab with the MatGraph toolkit [102]. Assume that all nodes are stationary and located randomly on a 200×200 m^2 or 800×800 m^2 area. The number of nodes ranges from 10 to 150. The transmission range is set to 100 or 400 meter depending on the experiment. Assume that each node has a dedicated antenna/beam for each neighbor, all nodes operate on the same frequency, and each link l operates at a maximum rate of C_l =10 Mbit/s. The number of flows $|\mathcal{F}|$ ranges from 5 to 75. For each flow, a source and destination nodes are randomly selected, and is routed over the shortest path. This section sets $\epsilon = 0.001$; equivalent to when the number of slots in $|B_n|$ is more than 1000. Notice that Algo-Fair rarely reaches this ϵ limit in the experiments, where the additional bandwidth provided to flows is negligible. Each result point is an average of 10 experimental runs on the same topology. Emphasize that the focus is on superframe construction and slot allocation; i.e., this chapter is not concerned with channel conditions. Note that any retransmissions can be carried out in subsequent slots for a given number of times.

This section evaluates Algo-Fair against six other designs, namely, Algo-2OL, Algo-1OL, Algo-2Flow, Algo-1Flow, A2Greedy and A1Greedy. The main difference among them is as follows. Algo-2OL, Algo-2Flow and A2Greedy apply Algo-2 [32], and Algo-1OL, Algo-1Flow and A1Greedy use Algo-1 [103]. To aid readability, it uses the label '1' or '2' to denote the scheduler in question, e.g., Algo-2OL and Algo-1OL correspond to Algo-2 [32] and Algo-1 [103], respectively. In experiments, they serve as example schedulers that do not consider flow fairness. Note that Algo-2Flow and Algo-1Flow emulate the behavior of the end-to-end water filling algorithm of

[104]. A key difference, apart from being distributed, is that the algorithm in [104] assumes a tree topology whose nodes form a matching in each time slot, i.e., one transmission/reception per node.

This section now briefly explains the six schedulers. Algo-2OL and Algo-1OL first generate a basic superframe before adding all possible opportunistic links into each slot. This means if all links of a flow are added as opportunistic links, then its rate will increase. In contrast, after generating a basic superframe, Algo-2Flow and Algo-1Flow increase the weight of flows iteratively. Specifically, they randomly pick a flow, and increase its weight by one. If this causes other flows' rate to reduce, they then revert the flow's weight to its previous value. Otherwise, they retain the flow's weight and move to the next flow. This process repeats until no flow weight can be increased. The last two algorithms, A2Greedy and A1Greedy, focus on throughput rather than fairness. They iteratively consider flows with increasing hop count, starting from the shortest, and generating a new superframe in each iteration to accomodate newly added flows. If a flow, say f_z, causes the rate of other flows to reduce, they revert to the superframe of the previous iteration, and f_z is removed from consideration.

In each experiment, the following metrics are collected:

- *Flow rates.* The final flow rate is equal to the minimum link rate along its path. This is computed using Equ. (4.1), where n_i^l is the number of slots assigned to link l for flow f_i, and $|S|$ is the superframe length,

$$r_i = MIN\left\{\frac{n_i^l}{|S|} |\ l \in f_i\right\} \tag{4.1}$$

- *Average total throughput.* This is simply the sum of all flow rates r_i divided by the number of simulation runs, which is 10.

The first experiment is over three 200×200 m^2 randomly generated topologies with 10, 15 and 20 nodes. In Figure 4.4, the minimum flow rate generated by

Algo-Fair is 1.67 Mbit/s when the number of nodes is 10. This is higher than the minimum rate generated by Algo-1OL, Algo-1Flow, A1Greedy, i.e., 1.25 Mbit/s, and A2Greedy, i.e., zero. The reason is because Algo-1 generates longer basic superframe lengths than Algo-2, and the minimum flow rate is decided by the basic superframe. A2Greedy starved two flows while allocating other flows with a rate of 2.5 Mbit/s. This is because A2Greedy assigns a higher priority to shorter flows. When the number of nodes is 15, the minimum rate generated by Algo-Fair is 1.43 Mbit/s. This is higher than the minimum rate, i.e., 0.83 Mbit/s, generated by Algo-1OL, Algo-1Flow and A1Greedy. Also observe that Algo-Fair equally allocates an additional 1.43 Mbit/s to two flows rather than allocating 1.43 Mbit/s to one flow as is the case with Algo-2Flow. When the number of nodes is 20, A2Greedy and A1Greedy starved one flow. Algo-1OL and Algo-1Flow ensure a minimum rate of 1.67 Mbit/s. This is lower than the minimum rate, i.e., 3.33 Mbit/s, generated by Algo-Fair because Algo-1 generates longer superframes than Algo-2. In this case, Algo-Fair equally allocates an additional 6.67 Mbit/s to four flows rather than allocating 6.67 Mbit/s to two flows. In contrast, Algo-2OL allocates an additional 6.67 Mbit/s to two flows and Algo-2Flow allocates 3.33 Mbit/s to one flow.

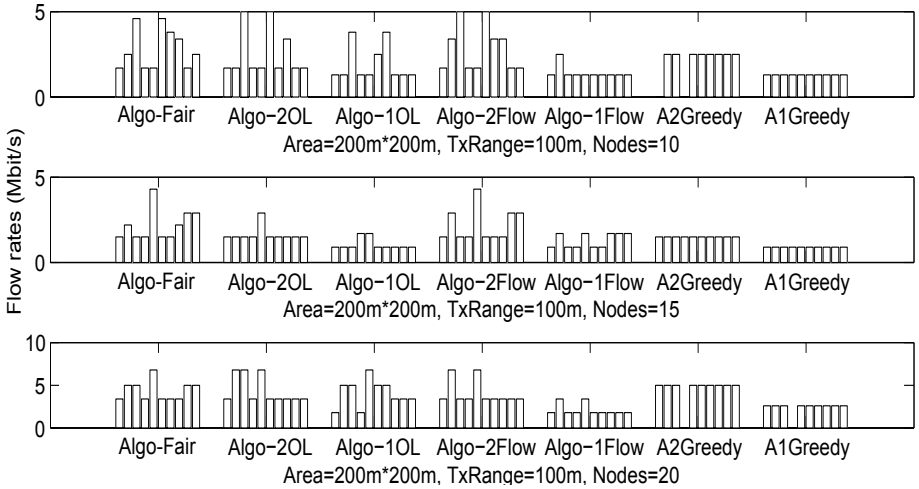


Figure 4.4: Allocated rates of ten flows on three topologies located in an 200×200 m^2 area with 10, 15 and 20 nodes

This section also compares the flow rates computed by all algorithms to the optimal MMF rate. This is carried out manually, and hence, is only possible for

small topologies. Consider Figure 4.3. All flows are assigned a weight of one, meaning each link's weight is equal to the number of traversing flows. Then the weight of a node is set to the sum of the maximum incoming link weight and the maximum outgoing link weight. In Figure 4.3, node E is the heaviest node because it has an incoming link with a weight of two and an outgoing link with a weight of two. Node E is thus marked as a bottleneck node, and link DE and EF are bottleneck links. In order to satisfy the no Mix-Tx-Rx constraint, each flow traversing link DE and EF can only receive a fair share of 1/4. Thus, the rate of flow f_1, f_3 and f_4 is 2.5 Mbit/s. We see that flows f_2 and f_1 share link BC. Flow f_1 is bottlenecked by link DE and EF. Thus, flow f_2 can use the remaining capacity of link BC, which is $1 - 1/4 = 3/4$. The flow f_2 will have a rate of $3/4 \times 1/2 = 3/8$ because node B and C need to receive and transmit data for flow f_2. Thus, the rate of flow f_2 is 3.75 Mbit/s. Recall that earlier in Section 4.3, using Algo-Fair, the following rates can be computed: $r_1 = 1/4$, $r_2 = 3/8$, $r_3 = 1/4$ and $r_4 = 1/4$. These rates are in fact optimal.

Table 4.3 shows the gap as compared to the optimal flow rates. We see that Algo-Fair has the lowest average gap, i.e., 0.21 Mbit/s and 0.14 Mbit/s, for the topology with 10 nodes and 15 nodes respectively. When the number of nodes is 20, Algo-Fair matches the optimal rates. The gap between the results of Algo-Fair and the optimal flow rates is due to two reasons. First, note that Algo-2 is a heuristic for the NP-complete, MAXCUT problem. Thus, it may not generate the shortest superframe for a given topology. In other words, the minimum rate received by flows is non-optimal. Second, as opportunistic links are added greedily, see *line 26* of Algorithm 2, the resulting allocation of opportunistic links may cause unnecessary augmentations.

This experiment studies increasing number of flows, i.e., 5 to 75, using a randomly generated topology on a 800×800 m^2 area with 75 nodes. ¿From Figure 4.5, we see that Algo-Fair generates the highest throughput, with an average throughput that is 8.3 Mbit/s higher than Algo-2Flow, and 130.5 Mbit/s higher than A1Greedy. When

Algorithm	Average Gap		
	Nodes=10	Nodes=15	Nodes=20
Algo-Fair	0.21	0.14	0
Algo-2OL	0.5	0.43	0.67
Algo-1OL	1.13	0.75	1.17
Algo-2Flow	0.33	0.21	0.67
Algo-1FLow	1.63	0.33	1.17
A2Greedy	1.33	0.43	0.5
A1Greedy	1.75	0.67	1.25

Table 4.3: Average gaps as compared to the optimal MMF

there are five flows, Algo-Fair, Algo-2OL and Algo-2Flow are both at 36 Mbit/s because their first two steps are the same. When there are 60 flows, Algo-Fair increases the average total throughput by more than 73.2% as compared to A2Greedy, and about 10.4% as compared to Algo-2OL and about 6.9% as compared to Algo-2Flow. When using A2Greedy, longer flows may starve. In Algo-2OL, the rate allocated to a link will increase significantly if this link is assigned several opportunistic slots. However, the rate of a flow is constrained by the minimum link rate along its path. The results generated by Algo-2Flow are very close to those of Algo-Fair when the number of flows is small, i.e., 5, 10, 15 and 20. Nevertheless, if two opportunistic flows, say f_i and f_j, share one link l, and l only has one opportunistic slot in the generated superframe, Algo-2Flow may not fairly increase the rates of the two flows. If Algo-2Flow first selects flow f_i and increases its weight by one, then flow f_j's weight cannot be increased without adding a new slot to link l. The rate allocated to flows will decrease if the superframe length increases. Thus, Algo-2Flow cannot fairly allocate additional bandwidth in some cases.

The last experiment studies the effect of node numbers and degrees on the average total throughput. The number of flows is fixed at 30. Both Figure 4.6 and 4.7 show that Algo-Fair generates the highest throughput. With varying node numbers, on average, the throughput of Algo-Fair is 11 Mbit/s and 102.6 Mbit/s higher than the throughput of Algo-2Flow and A1Greedy, respectively. As for different node degrees, Algo-Fair recorded an average that is 4.1 Mbit/s and 48.2 Mbit/s higher than Algo-2Flow and A1Greedy, respectively. In both experiments, when there is a higher number of nodes or node degrees, there are more links. Consequently we see

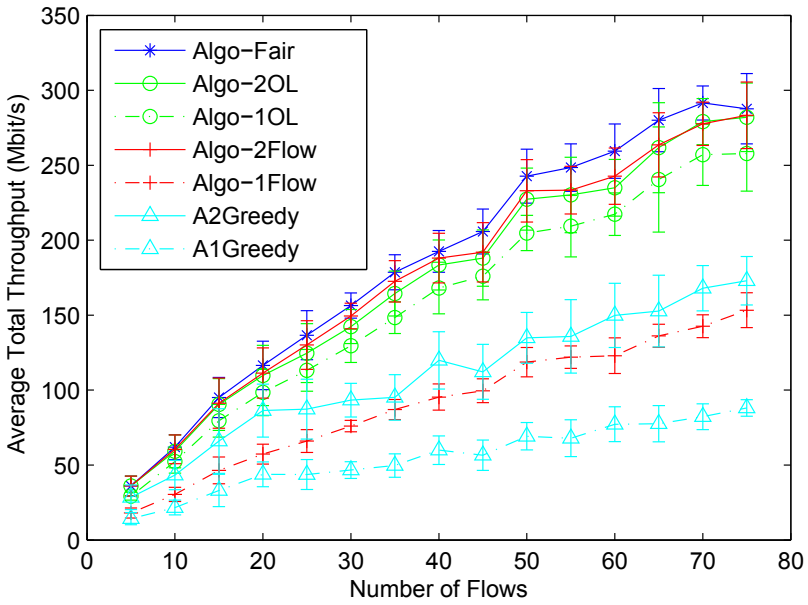


Figure 4.5: Average total throughput with different number of flows

a drop in the number of flows traversing each link. In contrast, the load on each link is higher when the number of nodes or node degrees is small. In this scenario, Algo-Fair will assign the same additional bandwidth to each contending flow. However, in the other six algorithms, one or more flows are starved. On the other hand, in the former case, as there are fewer flows on each link, they will receive a higher rate. We see that Algo-Fair also performs well as it maximizes the rate of each flow while the other six algorithms only maximize the rates of a subset of flows.

4.6 Conclusion

In summary, Algo-Fair has the following advantages: 1) it ensures all flows are allocated a minimum rate, meaning no flows starve, 2) it uses Algo-2 [32] to ensure the highest possible minimum rate, and 3) it employs an augmentation operation to ensure all opportunistic flows are assigned a fair increase in rate. The key novelty is step 3) where a novel augmentation step is used to exploit opportunistic slots. A similar step can be applied when deriving the superframe for other wireless systems. Compared with other designs, such as Algo-2OL, Algo-1OL, Algo-2Flow,

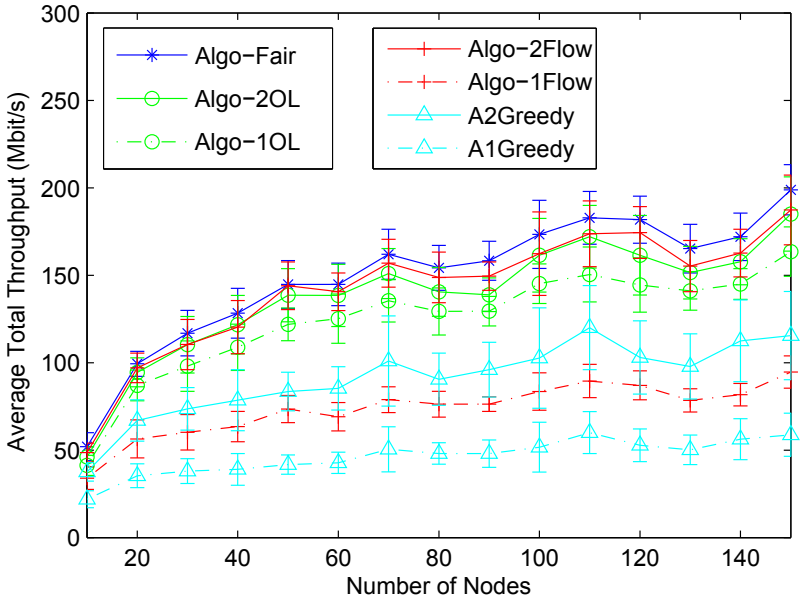


Figure 4.6: Average total throughput with different number of nodes

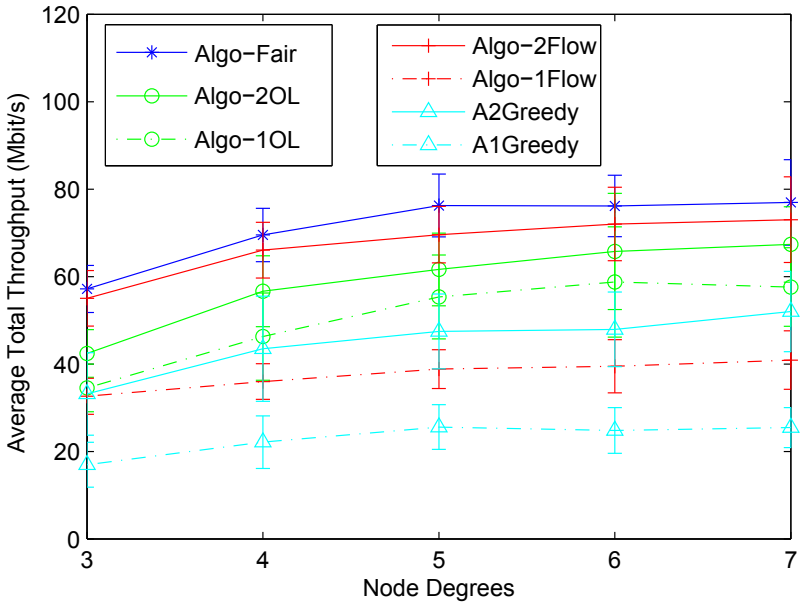


Figure 4.7: Average total throughput with different node degrees

Algo-1Flow, A2Greedy and A1Greedy, experiment results show Algo-Fair yields higher fairness in all tested scenarios.

A key limitation of the algorithms presented in Chapter 3 and 4 is that traffic demands are assumed to be fixed and known in advance. However, this assumption is not valid in practice. Traffic information is likely to vary and is generally random. This means deriving a superframe based on demands at a given time may lead to idle slots and severe packet delay. Moreover, it is impractical to recompute the superframe every time when traffic demand changes. Therefore, the next chapter will outline an approach that constructs a robust superframe for MTR WMNs with stochastic demands.

Chapter 5

Superframe Construction with Stochastic Demands

All existing MTR link schedulers assume fixed link load and traffic are known in advance. This information is then used to derive the shortest superframe. However, in practice, link load is likely to vary, meaning the computed schedule or superframe will be insufficient or have unnecessary idle times. In addition, in large-scale multi-hop WMNs, recomputing the superframe whenever the traffic changes is impractical; it requires frequent load information from nodes, and once a new superframe is installed, it may be dated. Apart from that, as a WMN will have peak and off-peak periods, different channel access mechanisms are required. When link load is low, random channel access is preferred because a link can relinquish the channel when it has finished transmission. In contrast, when link load is high, a TDMA schedule is ideal because it affords links collision-free channel access.

To this end, this chapter outlines a solution that combines the advantages of TDMA and random channel access. It presents a superframe with a scheduled and a random access part. Advantageously, the size of each part is adjustable according to random traffic demands. It is worth noting that in existing technologies such as

IEEE 802.15.3 [34], the *boundary* between the scheduled and random access part is fixed. Ideally, if the link load is high then the TDMA part should be sufficiently long to ensure little or no collision. Conversely, in scenarios where the load is low, the superframe will contain an elongated random access part to allow nodes to gain access when needed.

This chapter presents an approach based on Stochastic Programming (SP) [105] to size a superframe. Its key advantage is avoiding frequent superframe computation whenever the traffic demands change. It is the first to demonstrate a novel application of SP to generate a superframe that balances idle times and collisions. In addition, the proposed SP approach has a *control knob*, in the form of a penalty value, that allows a network operator to size both parts of the derived superframe according to offered traffic load. This chapter also shows that the proposed approach can be embedded within a binary search to obtain the best penalty value for a given probability of collisions and idle times. Lastly, this approach is general and can be used for example to size the contention access period (CAP) and channel time allocation period (CTAP) of IEEE 802.15.3 wireless networks [34]; similarly, it can be used in other forms of wireless networks. Numerical results show the efficacy of the proposed approach in reducing idle times and collisions given random demands.

5.1 Preliminaries

Consider a multi-hop WMN comprising of MTR-capable nodes as an arbitrary graph $G(V, E)$, where V denotes the set of nodes and E represents the set of directed links. Let $(u, v) \in E$ denote a directional link from router u to v. It has a normalized capacity of one. Let $L_{(u,v)} \in [0, 1]$ represent the load of link (u, v). Let $\mathbf{L}$ denote a column vector with $|E|$ elements that record the load of all links; i.e., $\mathbf{L} = \{L_{(u,v)} \mid (u, v) \in E\}$.

Let $\mathbf{A}$ be a $|E| \times N$ matrix that represents the collection of transmissions sets that can be activated in a given time instance. Specifically, each column represents

a set of non-conflicting links. The j-th column is denoted as $\mathbf{A}_{*j}$. Each element of column j, denoted as $a_{*j}^{(u,v)} \in \{0,1\}$, specifies whether link (u,v) is active in column j. As an example, if there is a column $[0101]^T$ corresponding to edge e_1, e_2, e_3, and e_4, then edge e_2 and e_4 can be activated together. The set of columns in $\mathbf{A}$ can be generated via an exhaustive search or using a heuristic. As an aside, for non-MTR WMNs, the matrix $\mathbf{A}$ represents the set of links that adhere to the protocol or physical interference model, in which case each column is an independent set [18]. In this case, each column is a maximum cut (max cut). As generating the max cut of a topology is a well known NP-complete problem, a heuristic called Algo-2 [103] is used. Briefly, Algo-2 creates a bipartite graph with maximal matching or a column of the matrix $\mathbf{A}$ by separating nodes into two sets: Set1 and Set2. Initially, Algo-2 includes all nodes in Set1 and sets Set2 to empty. It then moves a node from Set1 to Set2 if this increases the number of edges connecting nodes in Set1 and Set2. Upon processing all nodes, a max cut is generated and all links that span the two sets are removed from the topology. The next max cut can be obtained by repeating Algo-2 on the revised topology. The total number of max cuts generated is denoted as N. It is worthwhile noting that there is no need to generate all exponentially many max cuts. This is reasonable because the problem at hand is not to generate the maximum network capacity.

Let x_j denote the active time of the max cut $\mathbf{A}_{*j}$, where $0 \leq x_j \leq 1$. A TDMA schedule, comprising of the duration in which each set of links or a max cut is active, can then be represented as non-zero elements of the column vector $\mathbf{S} = [x_1, x_2, \ldots, x_N]^T$, where $\sum_{j=1}^{N} x_j \leq 1$.

In the random access part of the superframe, nodes employ Slotted Aloha for channel access. Notice that a more sophisticated MAC is not required because a node is able to receive from multiple neighbors simultaneously due to its MTR capability. Recall that a transmitting node will experience a collision only if at least one of its neighbors transmit to it at the same time. Notice that this is different to WMNs that assume the protocol interference model where multiple transmissions

from neighboring nodes will likely lead to collision. Recall that MTR capability allows a node to transmit/receive to/from all neighbors simultaneously.

5.2 Problem Definition

First, consider the scheduling problem with *deterministic* traffic demands: determine the shortest schedule period such that link loads are satisfied subject to link capacity constraint. Mathematically, the following Linear Program (LP) aims to minimize the schedule length,

$$\text{MIN} \sum_{j=1}^{N} x_j \tag{5.1}$$

$$\text{s.t.} \sum_{j=1}^{N} a_{*j}^{(u,v)} \times x_j \geq L_{(u,v)}, \quad \forall (u,v) \in E \tag{5.2}$$

$$\sum_{j=1}^{N} x_j \leq 1 \tag{5.3}$$

Constraint (5.2) ensures the total active time of an edge is sufficient to satisfy a link's load. The indicator variable $a_{*j}^{(u,v)}$ ensures only max cuts containing link (u, v) are included. Lastly, (5.3) limits the total active time of all max cuts to one. Notice that the demand, i.e., $L_{(u,v)}$, is fixed for each link. If the load of a link changes, its allocated transmission opportunities may be insufficient if the link's load becomes higher; i.e., constraint (5.2) is violated. Conversely, if the link is unable to use its allocated transmission time, i.e., there is less demand than originally predicted, the allocated time will be unused/wasted. Consequently, whenever there is a change in link load, the superframe or link schedule will have to be recomputed and re-transmitted to all nodes.

5.3 Approach

The objective of this chapter is to generate a superframe, see Figure 5.1, whereby links are provided with collision-free channel access as in the fixed demands case but sized accordingly to minimize idle times. Moreover, when the TDMA part is inadequate to meet a link's demand, the "overflow" traffic is transmitted in the random access part.

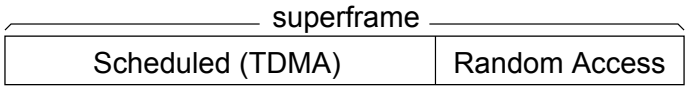


Figure 5.1: A superframe with scheduled and random access parts

This section proposes an approach that uses two-stage Stochastic Programming (SP). Briefly, in a two-stage SP, see [105] for more details, there are two sets of decision variables. In the first stage, the aim is to determine the best resource allocation that will minimize the expected recourse cost taken in the second stage. The second stage is then to minimize a given objective considering the first set of variables and a given random outcome. Mathematically, the first stage solves

$$\min_{x \in X} \{g(x) := c^T x + \mathbb{E}_\rho[Q(x, \xi)]\}, \tag{5.4}$$

and the second stage is to solve

$$Q(x, \xi) = \min_y \{q^T y \mid Tx + Wy \leq h\} \tag{5.5}$$

The first stage has decision variable x with cost c that is chosen in order to minimize the expected recourse cost of the second stage problem; note, y, with cost q, can be interpreted as any shortfall to be fulfilled given allocation x. The data used in the second stage, denoted as $\xi = (q, h, T, W)$, has one or more random variables governed by the probability distribution ρ. Each realization k of ξ, denoted as ξ_k, is called a scenario. Note, the expectation in (5.4) is carried out with respect to probability distribution ρ.

In the said link scheduling problem, the decision variables of the first stage correspond to the length of the scheduled part of the superframe, and *random* link load is in the second stage problem; i.e., the link load vector $\mathbf{L}$ is now random, and is denoted as ξ_k. The aim is to produce a superframe with a suitable x or TDMA schedule length that supports all possible random link-load vectors such that there is minimal idle time in the TDMA part, and a reasonable collision probability in the random access part. Thus, in the first stage, the formulation is

$$\text{MIN} \sum_{j=1}^{N} x_j + \mathbb{E}_\rho[Q(x, \xi_k)] \tag{5.6}$$

$$\text{s.t.} \sum_{j=1}^{N} x_j \leq 1 \tag{5.7}$$

For the second stage, given a scenario k, the formulation is

$$Q(x, \xi_k) = \text{MIN} \sum_{j=1}^{N} q^T y_j \tag{5.8}$$

$$\text{s.t.} \sum_{j=1}^{N} a_{*j}^{(u,v)} \times (x_j + y_j) \geq \xi_k^{(u,v)}, \quad \forall (u, v) \in E \tag{5.9}$$

$$\sum_{j=1}^{N} (x_j + y_j) \leq 1 \tag{5.10}$$

where $x_j, y_j \in [0, 1]$. Note, y_j can be interpreted as the amount of residual traffic after x_j of a link's demand has been serviced in the scheduled part. This residual or overflow traffic will have to be served in the random access part. Constraint (5.9) ensures the transmission opportunities for each link (u, v) determined in the first stage, i.e., the scheduled/TDMA part, plus the recourse to cover any shortfall are

at least $\xi_k^{(u,v)}$. The last constraint, i.e., (5.10), ensures the total superframe length remains less than or equal to one.

All links have the same penalty value q, see (5.8), in all scenarios. The value q has a significant impact on how the superframe is divided into scheduled and random access parts. If the penalty value is high, meaning overflow traffic is expensive, the resulting superframe will contain a longer scheduled part to minimize the objective function. However, if the penalty is small, the random access part will be longer; Section 5.4 will study the impact of q and outline a binary search method to determine the smallest q.

The formulated SP can be solved using any LP solver; see [105]. However, a key challenge is the exponential number of scenarios. For example, if the load of each link is discretized into K values, then there are $K^{|E|}$ scenarios. To this end, the Sample Average Approximation (SAA) method [105], a Monte-Carlo sampling technique, is employed to estimate the true expectation. Specifically, expression (5.6) can be revised to

$$\text{MIN} \sum_{j=1}^{N} x_j + \frac{1}{M} \sum_{k=1}^{M} Q(x, \xi^k) \tag{5.11}$$

As per [105], in evaluation, a value of M is chosen to ensure the estimated result has an accuracy of 2% with 90% confidence interval. Note that higher accuracy levels have only marginal effect on average idle times and average number of collisions experienced by nodes.

5.4 Evaluation

The experiments are conducted using CPLEX [106] and Matlab with the MatGraph toolkit [102]. Assume that 50 stationary nodes with MTR capability are randomly located on a 300×300 m^2 area. The transmission range is set to 50 meters. Assume

each node has a dedicated antenna/radio, see [25][21], for each neighbor. In the case of MIMO-based MTR WMNs, all nodes have the channel state information of their neighbors. Without loss of generality, experiments in this section assume uniform traffic distribution. Specifically, for a given scenario k, a random value is assigned to the components of ξ_k that is drawn from the range $[0, \lambda]$. Here, the maximum load or λ of links is obtained via an LP with an objective that aims to maximize the minimum active time of each link subject to constraint (5.7).

A key metric is the average collision in the random access part of the superframe. For a given node, its probability of collision is determined by its load, and the load of its neighbors. Observe that the link load in the random access part is due to the leftover traffic after the conclusion of the TDMA part. This can be called *overflow* traffic. For a link (u, v) in scenario $k \in [1, M]$, its overflow is calculated as $o^k_{(u,v)} = \xi^{(u,v)}_k - \sum_{j=1}^{N} a^{(u,v)}_{*j} \times x_j$, where $\xi^{(u,v)}_k$ is the link's demand; i.e., its total demand minus the proportion of demand transmitted in the TDMA part. Then, the average overflow on link (u, v) over all M scenarios is $\bar{o}_{(u,v)} = M^{-1} \sum_{k=1}^{M} o^k_{(u,v)}$.

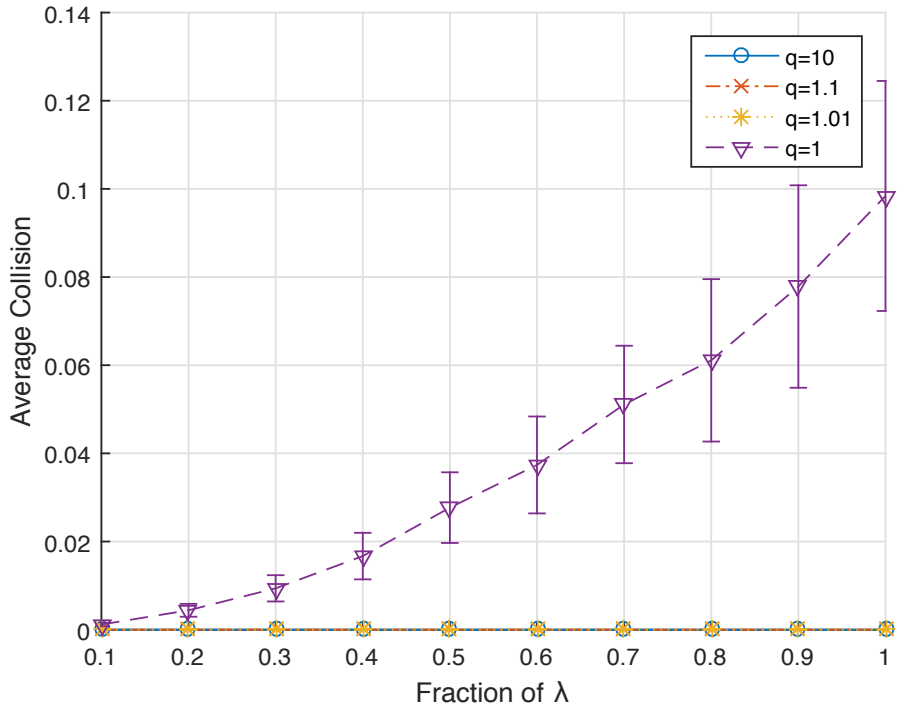


Figure 5.2: Average collision

The random access part for a given x has length $r = 1 - \sum_{j=1}^{N} x_j$. The probability

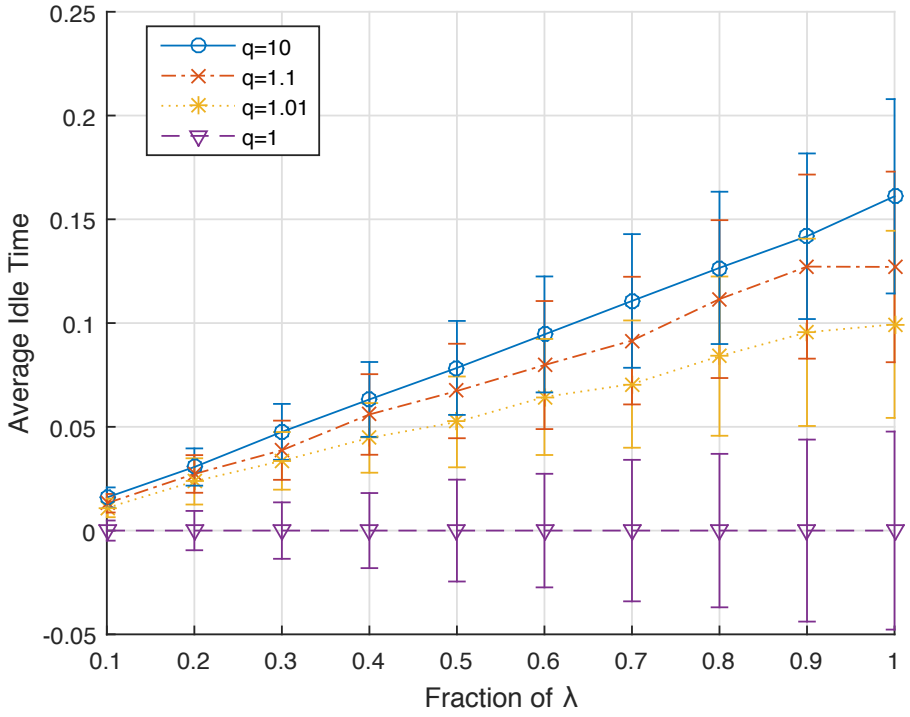


Figure 5.3: Average idle time with increasing traffic load

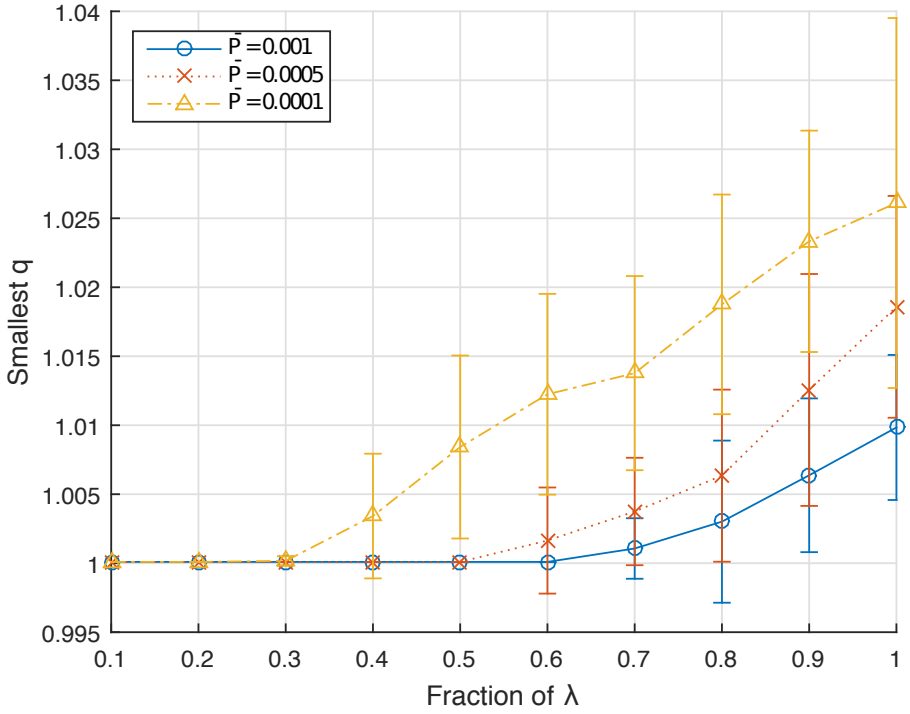


Figure 5.4: The smallest q with increasing traffic load for different $\bar{P}$ values; the accuracy of the binary search is set to 0.0001

that a link (u, v) transmits in the random access part is determined by its average overflow and the length of the random access part; i.e., $w_{(u,v)} = \frac{\bar{o}_{(u,v)}}{r}$. Let w_u be the probability that a node u transmits to any of its neighbors. Thus, $w_u = \max\{w_{(u,v)}\}$, where link (u, v) is any outgoing link of u. The probability that no neighbors transmit to u in the random access part is defined as N_u. Consider an incoming link (v, u) of node u. Thus, the probability that link (v, u) will not transmit to u is $1 - w_{(v,u)}$, and the probability that no neighbors transmit to u is $N_u = \prod_{v \in \mathcal{K}_u}(1 - w_{(v,u)})$, where $\mathcal{K}_u$ is a set containing neighbors of node u. Recall that MTR nodes experience collision only if one or more of their neighbors transmit to them whilst they are transmitting. This means the probability of collision at node u is $P_u = w_u \times (1 - N_u)$. The average collision over all nodes is thus $\bar{P} = |V|^{-1} \sum_{u \in V} P_u$. Another metric of interest is the average idle time. For a link (u, v) in scenario k, its idle time is $I^k_{(u,v)} = \sum_{j=1}^{N} a^{(u,v)}_{*j} \times x_j - \xi^{(u,v)}_k$. Thus, the average over M scenarios is $\bar{I}_{(u,v)} = M^{-1} \sum_{k=1}^{M} I^k_{(u,v)}$, and the average idle time over all $|E|$ links is $\bar{I} = |E|^{-1} \sum_{(u,v) \in E} \bar{I}_{(u,v)}$.

The effect of penalty values is presented below. Figure 5.3 shows that the value of $\bar{I}$ is always zero when $q = 1$ because the resulting superframe does not have a scheduled part. However, this penalty value makes the probability of collision increase with traffic load, see Figure 5.2, and is not suitable for use when the traffic load is high. On the other hand, if q is set to 10, the superframe will only contain a TDMA part that ensures $\bar{P}$ is close to zero when the traffic load increases. However, the resulting superframe has many wasted slots or idle time. From these figures, we also see that a small change in penalty value, e.g., $q = 1.1$ and $q = 1.01$, has a significant impact on idle time but less so on the probability of collision due to the use of MTR.

Given a traffic load and $\bar{P}$, the smallest q value that minimizes $\bar{I}$, or equivalently, the biggest random access part that meets the desired $\bar{P}$ value, can be computed using binary search. Initially, the start range is $[Q_{min}, Q_{max}]$; in the experiments, it is set to $[1, 100]$. In fact, according to Figure 5.2 and 5.3, any value above 10 is

suitable as Q_{max}. Then the experiment sets $q = (Q_{max} + Q_{min})/2$, and uses this q when generating a superframe. If $\bar{P}$ is not satisfied, then $Q_{min} = q$. Otherwise, $Q_{max} = q$ meaning the penalty value can be smaller, and the new range is $[Q_{min}, q]$. The process repeats until the difference $Q_{max} - Q_{min}$ is less than a given accuracy, e.g., 0.0001. From Figure 5.4, we see that in general a high penalty value is required for higher traffic loads or smaller $\bar{P}$ values. Note, the above process always has a solution because, in the worst case, the superframe only contains a scheduled part in which the probability of collision is zero.

5.5 Conclusion

The problem of sizing the TDMA and random access parts of a superframe given random traffic demands to ensure minimal idle times and probability of collisions remains open in multi-hop WMNs with MTR capability. To this end, this chapter outlines the first two-stage SP approach that allows network operators to derive a superframe that ensures the average probability of collisions is less than a given value and the idle time of TDMA slots is minimal. Numerical results over varying traffic loads confirm that a suitable superframe can be obtained via a control knob or penalty value that is determined using binary search.

A key observation is that the algorithm presented in this chapter only considers random link loads and deriving the corresponding link schedule. The next chapter will consider random end-to-end demands and deriving a suitable routing and link schedule that support these demands. In particular, it will propose a joint routing and scheduling approach that supports end-to-end random demands that are described by polyhedral model [87].

Chapter 6

Joint Routing and Scheduling with Random Demands

Existing MTR link schedulers and works that jointly consider routing and scheduling in wireless networks assume traffic demands are known in advance and are fixed. However, in practice, end-to-end traffic is likely to be uncertain and unknown [35]. Consequently, the computed superframe may have excessive idle slots or cause links to have insufficient capacity at times of peak demands. Moreover, uncertain demands may cause a network operator to compute and install a new routing and superframe frequently; this is likely to incur high signaling overheads, especially in large scale multi-hop WMNs. Past works have considered oblivious routing over WMNs; e.g., [93]. They aim to establish a static routing that minimizes the worst case congestion given random demands.

Different from past works, this chapter aims to derive a routing and superframe solution that supports *all* demands characterized by a polyhedral set [87]. A key problem is how to derive a suitable superframe that supports all possible demand values or traffic matrices (TMs) as defined by the given polytope. Another problem, which has an impact on the superframe, is routing. Specifically, it controls

the amount of traffic routed on a given path or links. This means the resulting superframe must allocate the correct number of slots to links to support demands. In other words, routing determines link load, which dictates the active time of links, and ultimately, the derived superframe length.

This chapter first formalizes the problem using a semi-infinite Linear Program (LP). Then, it presents a solution to the formulated LP. Specifically, it outlines a novel algorithm, named Algo-PolyH, that employs two LPs to determine the maximal capacity of each link for a given MTR WMN and a feasible TM in a given polyhedral model. It applies two other LPs to iteratively generate a robust routing and derive the shortest superframe that supports all TMs in the given polyhedral set. This fact is confirmed in the evaluation of Algo-PolyH in networks with varying number of degrees, number of flows, number of nodes and number of paths.

6.1 Preliminaries

A key advance in characterizing TMs in a manner that is amenable to computation is the polyhedral model [87]. Specifically, traffic demands are described by a set of linear inequalities. Consider the triangle topology shown in Figure 6.1. A possible set of inequalities for the two flows is as follows: $d_{AC} + d_{BC} \leq 0.6$, $d_{AC} \leq 0.4$ and $d_{BC} \leq 0.4$, where d_{AC} and d_{BC} are the demands for the flow emanating from node A and B, respectively; here, the values 0.6 and 0.4 are fractions of the unit capacity that represents the amount of traffic. Notice that the demand of both flows can take on a range of values. Given the above inequalities, the extreme points of the resulting polytope are $(0, 0)$, $(0.4, 0.2)$ and $(0.2, 0.4)$.

A key problem in this chapter is to find a suitable superframe that supports all TMs in a given polyhedral model. Routing has a significant impact on the resultant superframe length because it decides the amount of traffic forwarded on each given path or link, and the link loads determine the activation time of links. As shown in Figure 6.1, flow (A, C) and (B, C) have two and one possible path, respectively.

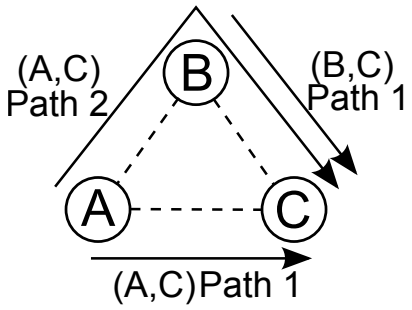
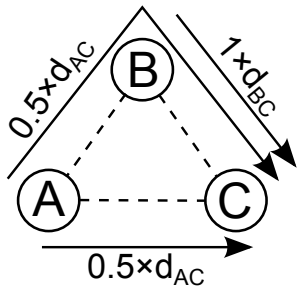


Figure 6.1: Example MTR WMN with two flows and given paths

Figure 6.2 and 6.3 show two example routings and the corresponding maximum link load. Recall that nodes support MTR, and thus link AB and AC can be activated together; so can links BC and AC. Thus, the following transmission sets or max cuts can be found: $\{AB, AC\}$ and $\{BC, AC\}$; these can then be used to construct a superframe. All that is required now is their active time.

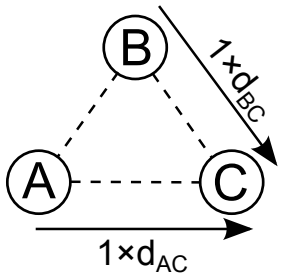
First assume that the demand of flow (A, C) is divided equally over Path 1 and 2, and all demand from flow (B, C) is routed over its only path; see Figure 6.2. Given the extreme point $(0.4, 0.2)$, the maximum load of link AB is $d_{AC} \times 0.5 = 0.4 \times 0.5 = 0.2$. The maximum load of link BC is $d_{AC} \times 0.5 + d_{BC} = 0.2 + 0.2 = 0.4$. On the other hand, given the extreme point $(0.2, 0.4)$, the maximum load of link AB and BC is $0.2 \times 0.5 = 0.1$ and $0.1 + 0.4 = 0.5$, respectively. From these two instances, to satisfy both extreme points, when deriving a superframe, the active time of links AB and BC must be ensured to support both extreme points. Thus, given this routing, the active time of max cuts $\{AB, AC\}$ and $\{BC, AC\}$ must be set to 0.2 and 0.5 respectively; the total superframe length is thus $0.2 + 0.5 = 0.7$. Now, assume a new routing whereby flow (A, C) routes all of its demand on Path 1; i.e., link AB has no traffic; see Figure 6.3. The maximum load of links BC and AC is 0.4 when the two extreme points are considered. Thus, max cut $\{BC, AC\}$ is only need to be activated for 0.4 unit time, resulting in a shorter superframe length. From these examples, jointly determining link schedule and routing is important to generate the shortest superframe for a given demand. Moreover, unlike prior works, the key challenge is that any solution must support all TMs that belong to a given

polyhedral set; specifically, TMs derived from any convex combinations of extreme points.



	Extreme Points	
	(0.4,0.2)	(0.2,0.4)
AB	0.2	0.1
BC	0.4	0.5
AC	0.2	0.1

Figure 6.2: First example routing. Each cell in the table shows the corresponding link load when $(d_{AC}, d_{BC}) = (0.4, 0.2)$ and $(d_{AC}, d_{BC}) = (0.2, 0.4)$



	Extreme Points	
	(0.4,0.2)	(0.2,0.4)
AB	0	0
BC	0.2	0.4
AC	0.4	0.2

Figure 6.3: Second example routing. Each cell of the table shows the corresponding link load when $(d_{AC}, d_{BC}) = (0.4, 0.2)$ and $(d_{AC}, d_{BC}) = (0.2, 0.4)$

Consider a single channel TDMA-based MTR WMN as an arbitrary graph $G(V, E)$, where V denotes the set of nodes, and E is the set of directed links between nodes. Let $(i, j) \in E$ represent a directional link from node i to j. Let c_{ij} denote the capacity of link (i, j). The set $F = \{(s, t) \mid s, t \in V, t \neq s\}$ is used to denote a set of flows where s and t are source and destination nodes, and $|F| \in [1, |V||V-1|]$. A TM is written as $d \in \mathbb{R}^{|F|}$ that records the demand information for each flow in F; e.g., d_{st} is the demand for flow from node s to t. Notice that d is technically a vector. However, the term *TM* is ubiquitous in the networking literature.

This chapter considers splittable flows and multi-path routing between each source and destination node pair. Thus, each flow is given a set of possible routing paths. In this chapter, a routing means the fraction of traffic routed on each link for a given flow $(s, t) \in F$. Let f_{ij}^{st} represent the fraction of traffic d_{st} routed on link (i, j). The values of all f_{ij}^{st} are recorded in a $|E| \times |F|$ matrix $\mathbf{f}$.

As mentioned above, a key consideration is uncertain traffic demands. Specifically, this chapter considers demands that can be described using a set of linear inequalities or equivalently, they exist in a polyhedron $\mathcal{D}$[87]. This polyhedral traffic model is defined as $\mathcal{D} = \{d \in \mathbb{R}^{|F|} \mid \mathbf{B}d \leq \alpha, d \geq 0\}$ where $\mathbf{B} \in \mathbb{R}^{K|F|}$, $\alpha \in \mathbb{R}^{K}$, and K is the number of constraints defining the polyhedron. The advantage of using a polyhedral model is that the proposed algorithm does not make any assumptions regarding traffic distribution. Indeed, there is no accepted traffic model for use in WMNs. In fact, the derived solution will ensure that the resulting superframe can support any value the traffic as long it is within the defined polytope.

A polyhedral model is also more practical as it does not require continuous monitoring of flows to characterize their traffic distribution. Indeed, a polyhedral model is particularly suited for WMNs because a mesh router tends to have multiple associated clients [1], from which it aggregates all traffic and direct them to one or more gateways or other mesh routers. Figure 6.4 shows a typical WMN. A network operator can obtain the maximum outgoing/incoming traffic from/to each mesh router via the Simple Network Management Protocol (SNMP). After that, a network operator can then define a special case/instance of the polyhedral model called *Hose* [88]. Formally, the formulation is showing below

$$\sum_{t:(s,t)\in F} d_{st} \leq C_s^+, \quad \forall s \in V \tag{6.1}$$

$$\sum_{s:(s,t)\in F} d_{st} \leq C_t^-, \quad \forall t \in V \tag{6.2}$$

where C_s^+ and C_t^- denote the outgoing and incoming traffic bounds for node s and t respectively, for all flows (s,t). As an example, reconsider the example discussed above. There are two flows (A,C) and (B,C) in the topology shown in Figure 6.4. The maximum outgoing traffic from both node A and B is constrained to 0.4, and the total incoming traffic of node C is 0.6. Then the following inequalities define the *Hose* model: $d_{AC} \leq 0.4$, $d_{BC} \leq 0.4$ and $d_{AC} + d_{BC} \leq 0.6$.

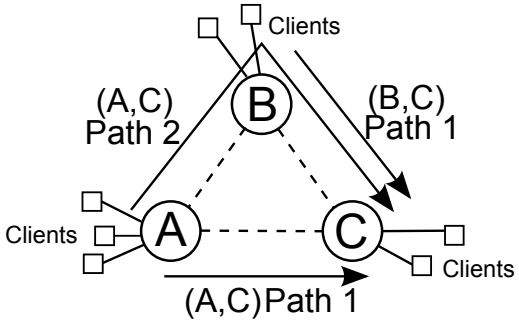


Figure 6.4: An example topology

Let a $|E| \times N$ matrix $\mathbf{A}$ represent the collection of N transmission sets that can be activated in a given time instance. Each column of $\mathbf{A}$ represents a set of non-conflicting links. The n-th column is denoted as $\mathbf{A}_{*n}$, where $n \in [1, N]$. Each element of column n or transmission set is a binary value, denoted as $a^{ij}_{*n} \in \{0, 1\}$, that indicates whether link (i, j) is active. Notice that the columns of $\mathbf{A}$ can be generated via an exhaustive search or using a heuristic algorithm. It is worth noting that for non-MTR WMNs, each column of matrix $\mathbf{A}$ represents an independent set that adheres to the protocol or physical interference model [18]. Consequently, the approach presented in this chapter is also applicable to other forms of wireless networks. For MTR WMNs, each column is a max cut of a topology; note that determining the max cut of a graph is a well-known NP-complete problem [99]. In order to generate the columns of $\mathbf{A}$, a heuristic algorithm called Algo-2 [32] is used to create N max cuts. As shown in [32], Algo-2 has a time complexity of $O(|V|^2)$. The reason of using a heuristic is because the maximum number of max cuts in a topology with $|E|$ links is $2^{|E|} - 1$. As shown in Section 6.4, this has a significant impact on computation time when the network is large. Hence, this chapter resorts to using a heuristic to generate the max-cuts for large networks. It is worth noting that the proposed algorithm works regardless of how the matrix A is generated; i.e., brute force or via a heuristic.

Briefly, given an arbitrary graph, it first creates a maximal bipartite graph by separating nodes into two sets: Set1 and Set2. Initially, Algo-2 includes all nodes into Set1, and initializes Set2 to empty. It then moves a node from Set1 to Set2

if this increases the number of edges connecting nodes in Set1 and Set2. It does this by selecting the node with the highest δ value in Set1, and moves it into Set2, where the δ value of node A is the difference between the total number of links from node A to other nodes in Set1 and the total number of links from nodes in Set2 to node A. Algo-2 stops moving nodes when the δ value of all nodes is less than or equal to zero. Upon processing all nodes, a max cut is found and all links that span the two sets are removed from the topology. The next max cut can be obtained by repeating Algo-2 on the revised topology. It is worthwhile noting that there is no need to generate all possible max cuts, for which there are exponentially many, for a given MTR WMN. In addition, $|E|$ columns are also added into the matrix $\mathbf{A}$; each of which contains one active link. These additional columns are used when only one link needs extra activation time.

Assume time is divided into slots. Each set of non-conflicting links or a max cut is active in one slot. A superframe is comprised of several time slots. Let x_n denote the active time of column n in matrix $\mathbf{A}$, where $x_n \in [0, 1]$. A TDMA schedule can then be represented as the column vector $\mathbf{S} = [x_1, x_2, \ldots, x_N]^T$, where $\sum_{n=1}^{N} x_n \leq 1$; the sum on the LHS is referred to as the *superframe length*.

6.2 Problem Definition

The problem is to determine a routing and the shortest superframe such that all TMs in $\mathcal{D}$ are satisfied subject to flow conservation and link capacity constraints. This joint routing and scheduling problem can be formulated as a semi-infinite LP,

$$\underset{f,x}{\text{MIN}} \sum_{n=1}^{N} x_n \tag{6.3}$$

$$\text{s.t.} \sum_{j:(i,j)\in E} f_{ij}^{st} - \sum_{j:(j,i)\in E} f_{ji}^{st} = \tag{6.4}$$

$$\begin{cases} 1 & i = s \\ -1 & i = t \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in V, \forall (s,t) \in F$$

$$\sum_{n=1}^{N} a_{*n}^{ij} x_n = c_{ij}, \qquad \forall (i,j) \in E \tag{6.5}$$

$$\sum_{(s,t)\in F} d_{st} f_{ij}^{st} \leq c_{ij}, \quad \forall d \in \mathcal{D}, \forall (i,j) \in E \tag{6.6}$$

$$\sum_{n=1}^{N} x_n \leq 1 \tag{6.7}$$

Constraint (6.4) is the standard flow conservation constraint. Equ (6.5) defines the capacity of a link as a sum of the active time of transmission sets in which a link belongs to. Constraint (6.6) ensures the active time of max cuts is sufficient to meet link demands. Lastly, constraint (6.7) ensures the total active time of all max cuts is one.

Notice that there are an infinite number of independent TMs in the polyhedron $\mathcal{D}$, meaning there are infinitely many constraints of type (6.6). Fortunately, every polyhedron has a finite but exponential number of *extreme points* [87]. Thus, for an arbitrary polyhedron $\mathcal{D}$, constraint (6.6) can be replaced by,

$$\max_{d\in\mathcal{D}} \left\{ \sum_{(s,t)\in F} d_{st} f_{ij}^{st} \right\} \leq c_{ij}, \quad \forall (i,j) \in E \tag{6.8}$$

Given (6.3)–(6.5), (6.7), and (6.8), the problem calls for a routing and a superframe that supports all TMs in $\mathcal{D}$.

6.3 Approach

This section proposes an iterative joint routing and scheduling heuristic algorithm called Algo-PolyH. It is based on the following general idea. It firstly generates

a routing and a superframe given a feasible TM in $\mathcal{D}$. Then it checks whether there exist other TMs in $\mathcal{D}$ that cannot be supported by the current routing and scheduling. If the answer is no, it means the routing and scheduling can support all TMs in the given polyhedral traffic model, and the algorithm ends. Otherwise, such TMs are added into the set $\mathcal{T}_k$, which contains all discovered extreme points of $\mathcal{D}$ up to iteration k. The process then repeats where it searches for a new routing and minimal superframe length that supports all TMs in $\mathcal{T}_k$.

Algo-PolyH relies on four key LPs: LP-CAPACITY, LP-INIT-TM, LP-MAIN and LP-NEWTM. Briefly, (i) LP-CAPACITY aims to maximize the minimal link capacity in an arbitrary MTR WMN with given max cuts and returns the maximal capacity of each link; let c_{ij}^* be the maximal capacity of link (i, j), and the column vector $\mathbf{C}$ of size $|E|$ records the maximal link capacity of each link, (ii) LP-INIT-TM aims to find a feasible TM in $\mathcal{D}$ while considering the maximal link capacity generated by LP-CAPACITY, (iii) LP-MAIN is used to generate a robust routing and scheduling that supports all TMs found thus far, and (iv) LP-NEWTM checks whether there exists TMs in $\mathcal{D}$ that cannot be supported by the routing and scheduling generated by LP-MAIN.

Next, each of the aforementioned LPs is formally defined. Given an arbitrary MTR WMN and its max cuts in $\mathcal{A}$, LP-CAPACITY is used to determine the maximum possible capacity attainable by each link in one unit time. It is worth noting that this LP, which produces the set $\mathbf{C}$, does not consider traffic demand. Formally, we have

$$\underset{x}{\text{MAX}} \quad \min_{(i,j)\in E} c_{ij} \tag{6.9}$$

$$\text{s.t. } (6.5)(6.7)$$

Algo-PolyH requires an initial TM. To this end, LP-INIT-TM aims to find a maximal TM d^0 within the given polyhedra model that meets the maximum capacity c_{ij}^* of each link; note, $c_{ij}^* \in \mathbf{C}$ is the solution returned by LP-CAPACITY. Initially,

all $(s,t) \in F$ are set to use only the shortest path, meaning the fraction of traffic is one on this path and zero for other possible paths; i.e., each link (i,j) on the shortest path of flow (s,t) has $f_{ij}^{st} = 1$. Note, this routing will be adjusted later by LP-MAIN. Formally, LP-INIT-TM is defined as follows,

$$\underset{d \in \mathcal{D}}{\text{MAX}} \sum_{(s,t) \in F} d_{st} \tag{6.10}$$

$$\text{s.t.} \sum_{(s,t) \in F} d_{st} f_{ij}^{st} \leq c_{ij}^{*}, \quad \forall d \in \mathcal{D}, \forall (i,j) \in E \tag{6.11}$$

LP-MAIN is explained as follows. The goal of LP-MAIN is to find a routing and the shortest superframe that supports all TMs found up to the $(k-1)$-th iteration. This means the resulting solution is *robust* against all possible TMs in $\mathcal{T}_{k-1}$. Then, LP-MAIN is,

$$\underset{f,x}{\text{MIN}} \sum_{n=1}^{N} x_n \tag{6.12}$$

$$\text{s.t.} \sum_{(s,t) \in F} d_{st} f_{ij}^{st} \leq c_{ij}, \quad \forall d \in \mathcal{T}_{k-1}, \forall (i,j) \in E \tag{6.13}$$

(6.4)(6.5) and (6.7)

Lastly, given a routing and superframe, Algo-PolyH needs to know whether there is an unsupported TM in $\mathcal{D}$. Such a TM exists if a link requires more active time in order to support the new TM. Let r_n denote the additional active time added to the transmission set n of the given superframe, where $r_n \geq 0$ and $n \in [1, N]$. An LP is then used, called LP-NEWTM, to check for the given routing and superframe, and a link (i,j), the maximum aggregated demand traversing the link. Formally, the formulation is,

$$\underset{d \in \mathcal{D}, r}{\text{MAX}} \sum_{(s,t) \in F} d_{st} f_{ij}^{st} \tag{6.14}$$

$$\text{s.t.} \quad \sum_{(s,t)\in F} d_{st} f_{ij}^{st} \leq \sum_{n=1}^{N} a_{*n}^{ij}(x_n + r_n), \quad \forall d \in \mathcal{D} \tag{6.15}$$

Observe that if any r_n is non-zero, then the superframe needs to be expanded to accommodate the maximum demand. This implies there exists an unsupported TM. Conversely, if r_n is zero, then the superframe and routing is sufficient. Consequently, if r_n is zero for all links, then we have found a solution that supports all demands in $\mathcal{D}$.

This section now presents the details of Algo-PolyH. It proceeds according to two phases: *initialization* and *main*. From Algorithm 3, Algo-PolyH first sets k to zero and $\mathcal{T}_k$ to empty, see *line 1*. It then solves LP-CAPACITY to obtain the maximal possible capacity of all links; see *line 2*. Next, Algo-PolyH solves LP-INIT-TM with $\mathbf{C}$ as input to find a feasible TM d^0 in $\mathcal{D}$. The resulting d^0 is then included into $\mathcal{T}_k$; see *line 3-4*.

At this point, a starting TM is found and Algo-PolyH is ready to determine the best routing and superframe that supports TMs in $\mathcal{T}_k$. In iteration k, Algo-PolyH solves LP-MAIN to determine a routing $\mathbf{f}$ and minimal superframe $\mathbf{S}$, see *line 9*. If LP-MAIN returns *infeasible*, then polyhedron $\mathcal{D}$ cannot be supported and Algo-PolyH exits; see *line 10-13*. Otherwise, with $\mathbf{f}$ and $\mathbf{S}$ in hand, Algo-PolyH calls LP-NEWTM to determine whether all demands in $\mathcal{D}$ are supported. For each link in E, it solves LP-NEWTM to check if the superframe needs to be extended to support a new $d^k \in \mathcal{D}$. If at least one r_n is non-zero and d^k is not in $\mathcal{T}_k$, Algo-PolyH adds d^k into $\mathcal{T}_k$ and moves to the next link; see *line 14-19*. After solving LP-NEWTM for all links, if no new TM is found, Algo-PolyH returns $\mathbf{f}$ and $\mathbf{S}$ as the final solution; see *line 20-23*. Otherwise, it seeks a new routing and superframe that supports all TMs in $\mathcal{T}_k$ by calling LP-MAIN in next iteration.

A few remarks regarding convergence. Algo-PolyH will terminate because of the following facts. First, if T_{k-1} contains a TM in which there is no routing or a link schedule, i.e., constraints (6.4), (6.5), (6.7) and (6.13) are not satisfied, then it will

Algorithm 3: Algo-PolyH

input : $G(V, E)$, $\mathbf{A}$, $\mathcal{D}$ and all possible paths for each flow
output: The fraction of traffic $\mathbf{f}$ and schedule $\mathbf{S}$

`// Phase 1: Initialization`
$k \leftarrow 0$, $\mathcal{T}_k \leftarrow \emptyset$
$\mathbf{C} \leftarrow$`Solve` LP-CAPACITY(G, $\mathbf{A}$)
$d^0 \leftarrow$`Solve` LP-INIT-TM($\mathcal{D}$, $\mathbf{C}$)
$\mathcal{T}_k \leftarrow d^0$
`// Phase2: Main`
Flag$\leftarrow$TRUE
while *Flag* **do**
  $k \leftarrow k + 1$
  $\mathcal{T}_k \leftarrow \mathcal{T}_{k-1}$
  $[\mathbf{f}, \mathbf{S}] \leftarrow$`Solve` LP-MAIN($\mathcal{T}_{k-1}$, $\mathbf{A}$)
  if *LP-MAIN is infeasible* **then**
    Flag$\leftarrow$FALSE
    Return Error('$\mathcal{D}$ is not supported')
  end
  for $e \in |E|$ **do**
    $[r_n, d^k] \leftarrow$`Solve` LP-NEWTM($\mathcal{D}$, $\mathbf{f}$, $\mathbf{S}$, e)
    if *any* $r_n > 0$ *AND* $d^k \notin \mathcal{T}_k$ **then**
      $\mathcal{T}_k \leftarrow \mathcal{T}_k \cup d^k$
    end
  end
  if $\mathcal{T}_k = \mathcal{T}_{k-1}$ **then**
    Flag$\leftarrow$FALSE
    Return ($\mathbf{f}$, $\mathbf{S}$)
  end
end

terminate with 'D is not supported (see line 12). Second, an LP-solver is guaranteed to return a TM that maximizes the objective (6.14). This is because the feasibility region of LP-NEWTM(), as well as LP-MAIN, is a polytope with finite number of extreme points. Also notice that the decision variable r_n is unbounded, meaning it can be made as large as possible to satisfy constraint of type (6.15).

The run time complexity of Algo-PolyH is governed by the formulated LPs; all of which can be solved in polynomial time [107]. Note, although the Simplex algorithm is known to have a worst case exponential running time, in practice and on average, it runs in polynomial time; e.g., as noted in [108], the run time of an LP solver is proportional to $\mathcal{O}(v^2 m)$, where v and m are the number of decision variables and constraints, respectively.

The computation time of Algo-PolyH is governed by the size of LP-CAPACITY, LP-INIT-TIM, LP-MAIN and LP-NEWTM. Henceforth, the remainder of this section presents the number of decision variables and constraints of these LPs. These facts will be useful when the computation time of Algo-PolyH is discussed in Section 6.4. This section will omit the proof of Proposition 10 and 13 as they can be readily ascertained by inspecting the corresponding LP formulation.

Proposition 10. *LP-CAPACITY has N decision variables and one constraint.*

Proposition 11. *LP-INIT-TM has $|F|$ decision variables and $|E|+K$ constraints.*

Proof. The decision variable is d_{st}, which is the demand for flow (s,t). There are $|F|$ flows. There is a corresponding constraint (6.11) for each link. Recall that the number of constraints defining the polyhedron is K. Thus, LP-INIT-TM has $|E|+K$ constraints. □

Proposition 12. *LP-MAIN has $N + |F| \times |E|$ decision variables and $|V| \times |F| + |E| \times |\mathcal{T}_{k-1}| + 1$ constraints.*

Proof. The decision variables in LP-MAIN is x_n and f_{ij}^{st}. The number of decision variables in LP-MAIN is $N + |F| \times |E|$. Now consider constraint (6.4). For each

node, Algo-PolyH needs to ensure the flow conservation constraint is satisfied for each flow. For each TM in $\mathcal{T}_{k-1}$, it needs $|E|$ constraints of type (6.13). Thus, the total number of constraints is $|V| \times |F| + |E| \times |\mathcal{T}_{k-1}| + 1$. □

Proposition 13. *LP-NEWTM has* $|F|+N$ *decision variables and* $1+K$ *constraints.*

As shown by Proposition 10 to 13, the running time of Algo-PolyH increases with the number of nodes $|V|$, number of links $|E|$, number of flows $|F|$ and number of max-cuts N. Indeed, as shown later in Section 6.4, the computation time increases with these parameters. Fortunately, Algo-PolyH can be applied to obtain a solution to networks of reasonable sizes; e.g., a solution for a 40 nodes MTR WMN can be obtained within 80 seconds. Moreover, a network operator does not need to re-compute the current routing and schedule as long as traffic varies within the defined polyhedral model; i.e., the traffic does not exceed the bound of constraints (6.1) and (6.2) or there are no new flows. Another remark is that the solution returned by Algo-PolyH is not necessarily unique. This is because an LP may have multiple optimal solutions [107]; this will depend on the set of constraints or geometry of the feasibility region. Apart from that, the solution is likely to change for different sets of paths or max cuts. This is because different paths or max cuts provide alternate routings and schedules.

6.4 Evaluation

The experiments are conducted using CPLEX [106] and Matlab with the MatGraph toolkit [102]. Assume all nodes are stationary and randomly placed on a 150×150 m^2 area. The transmission range is set to 50 meters. This results in a connected WMN with multiple hops. If the transmission range is too small, the generated topology will not be connected. On the other hand, if the range is too large, the topology will become a complete graph. Note, a similar effect can be achieved by changing the deployment area and adjusting the transmission range accordingly. Assume each

node has a dedicated antenna/beam for each neighbor. In the case of a MIMO-based MTR WMN, all nodes are assumed to have the channel state information of their neighbors. This section studies the number of flows, number of available paths for each flow, number of nodes and varying node degrees. The number of flows $|F|$ ranges from 5 to 40. As mentioned in Section 6.1, this chapter considers multi-path routing and splittable flows. Thus, each flow is given a set of available paths. Moreover, this section only considers paths with length, in terms of hops, that is less than or equal to the network diameter. The number of available paths ranges from 1 to 12. The number of nodes ranges from 5 to 40. This is reasonable and agrees with WMN deployments to date. For example, in [21], the authors document their experiences with a MTR WMN consisting of 32 nodes. The well-known MIT Roofnet WMN has 38 nodes [109]. Thus, this section only experiments with up to 40 nodes. The number of node degrees ranges from 3 to 7. Recall that this work focuses on routing and scheduling; channel conditions are not a concern.

The experiments in this section consider the *Hose model* [88]; a special instance of the polyhedral model. Thus, the set of linear inequalities bounds the outgoing and incoming traffic of each node. Without loss of generality, the Hose model is assumed to be symmetric, meaning a node's outgoing and incoming bounds are the same. Specifically, the model can be written as $C_s^+ = C_s^-$; this case can be denoted as $C_s^\pm$ where $s \in V$. In order to ensure the given polyhedral model can always be supported, the traffic load of links must be less than links' capacity. Thus, the bound $C_s^\pm$ is set to $1/|F| \times \min\{c_{ij} \mid (i,j) \in E\}$, where $s \in V$. It is worthwhile noting that this chapter is not concerned with determining the maximum capacity of an MTR WMN or the admissibility of a polyhedral set. In practice, the polyhedral set is given, which Algo-PolyH will then use to generate a robust routing and superframe.

In each experiment, the following metrics are collected:

- *Superframe length.* This records the average superframe length over 20 simulation runs. In each simulation run, the superframe length is the sum of x_n, where $n \in [1, N]$.

- *Computation time.* This is the average time required by Algo-PolyH to generate a robust routing and derive the shortest superframe. The experiments in this section are conducted on a computer with an Intel Core i7 with 6 GB RAM.

In the sequel, only results for Algo-PolyH are presented. This is because there are no other works, and Algo-PolyH addresses a new problem.

The first experiment studies how the number of flows impacts the superframe length and computation time. Assume 30 stationary nodes deployed on a square area. Each flow is routed over at most seven available paths. From Figure 6.5, as expected, we can see that the superframe length reduces with increasing number of flows. This is because $C_s^{\pm}$ is set to $1/|F| \times \min\{c_{ij} \mid (i,j) \in E\}$. For the same topology, the value of $\min\{c_{ij}\}$ will not change. Thus, the traffic bound $C_s^{\pm}$ reduces when the value of $|F|$ increases. In other words, the amount of traffic in the network reduces such that the total superframe length becomes short. Referring to Figure 6.6, the computation time increases with increasing number of flows. As discussed in Section 6.3, $|F|$ has an impact on the number of decision variables in LP-INIT-TM, LP-MAIN and LP-NEWTM as well as the number of constraints in LP-MAIN.

This next experiment considers the number of available paths. The number of flows is fixed at 20. The topology contains 30 stationary nodes. Figure 6.7 shows that the superframe length reduces with increasing number of available paths. As there are more paths, Algo-PolyH has more routing options. Thus, the probability of generating a bottleneck link reduces, which results in a shorter superframe length. From Figure 6.8, the computation time increases with the number of available paths. This is because a flow is able to split its traffic into more paths such that the number of links with non-zero load increases. Thus, LP-MAIN has more non-zero decision variables and constraint of type (6.13). The number of times that Algo-PolyH calls LP-NEWTM is also decided by the number of links with non-zero load. This explains why Algo-PolyH needs a longer time to find a robust solution.

The third experiment studies node numbers. The number of flows is fixed at 20,

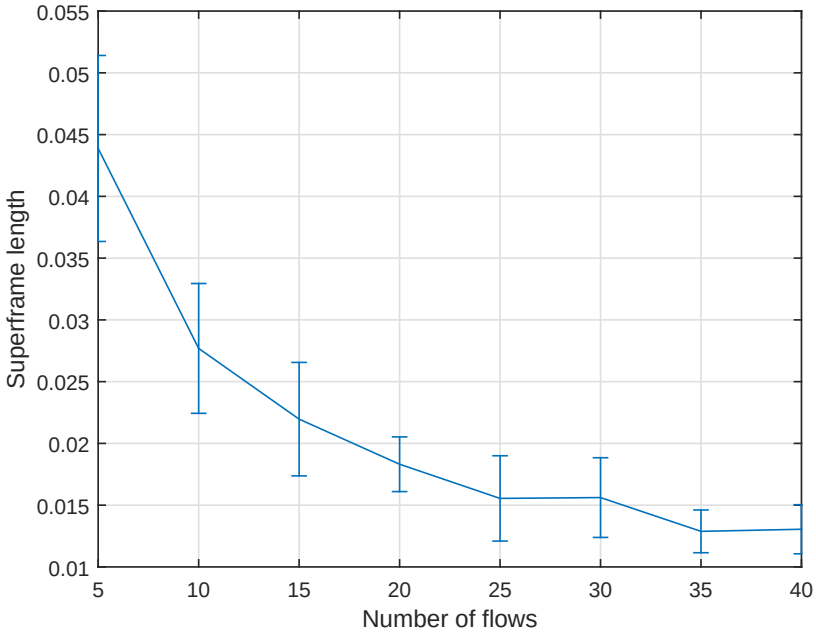


Figure 6.5: Average superframe length with different number of flows

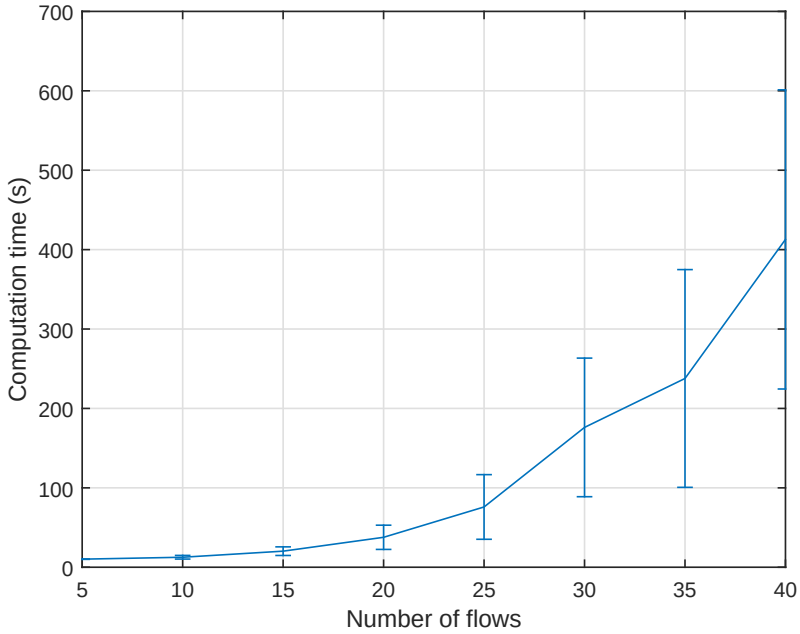


Figure 6.6: Average computation time with different number of flows

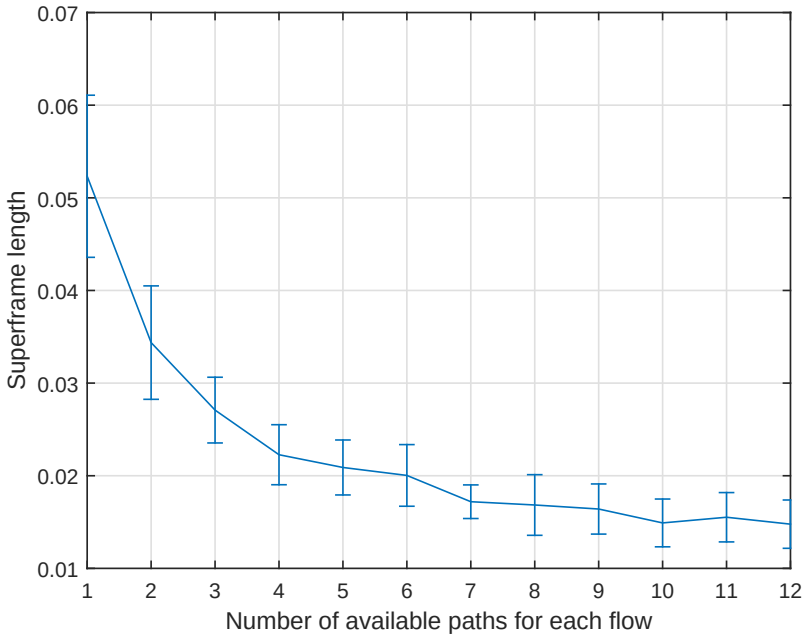


Figure 6.7: Average superframe length with different number of available paths

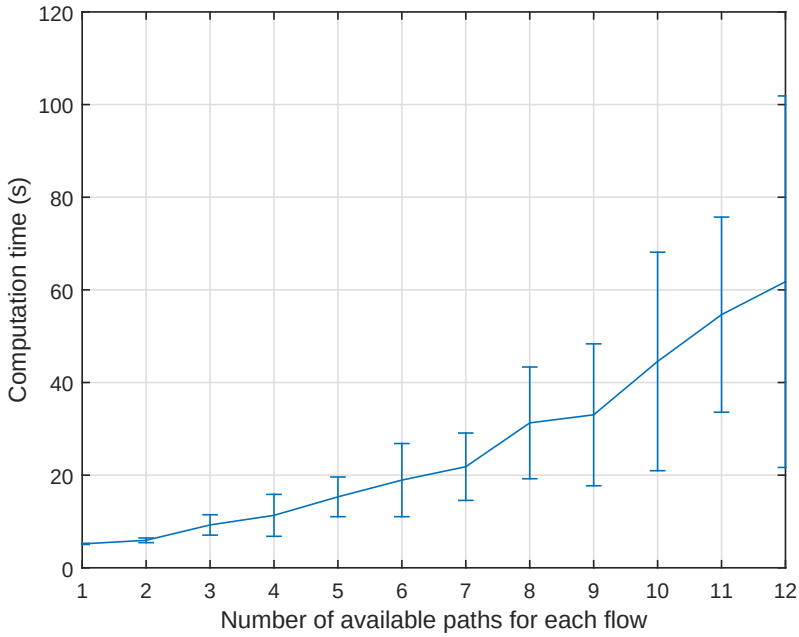


Figure 6.8: Average computation time with different number of available paths

and each flow is given seven available routing paths. Initially, a topology is generated randomly with five nodes. Then, the number of nodes increases by randomly adding new nodes to the same topology. From Figure 6.9, we can see that the resulting superframe length reduces when there are more nodes. This is because the capacity of an MTR WMN reduces with additional nodes. Recall that $C_s^{\pm}$ is set to $1/|F| \times \min\{c_{ij} \mid (i,j) \in E\}$, where the value $|F|$ is fixed in this experiment. As the number of nodes increases, each link (i,j) has more neighboring links that may conflict with it. Thus, the value of $\min\{c_{ij}\}$ reduces with increasing number of nodes; the value of $C_s^{\pm}$ also becomes smaller. From Figure 6.10, we can see that Algo-PolyH requires a longer computation time when there are more nodes. This is because the number of links is proportional to the number of nodes. Consequently, more max cuts are needed to ensure every link is contained in at least one max cut. In other words, the value of N increases. As mentioned in Section 6.3, N decides the number of decision variables in LP-CAPACITY, LP-MAIN and LP-NEWTM. In addition, the number of nodes $|V|$ decides the number of constraints (6.4). LP-MAIN requires a longer computation time when there are more nodes.

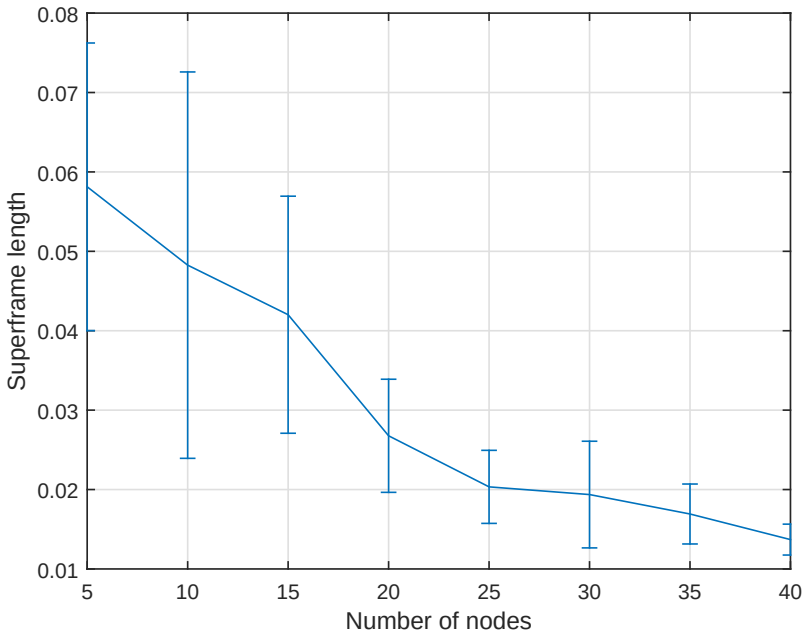


Figure 6.9: Average superframe length with different number of nodes

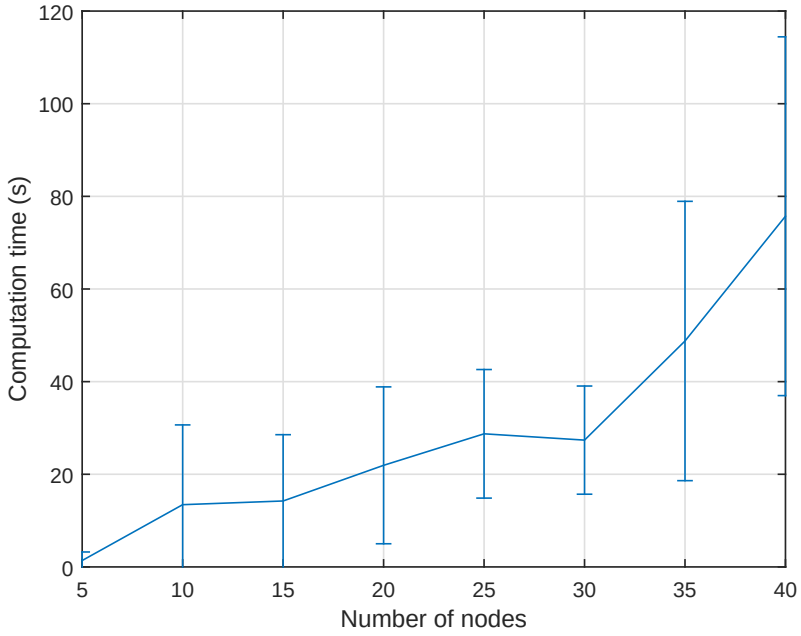


Figure 6.10: Average computation time with different number of nodes

Lastly, this section studies the impact of node degrees. The number of nodes is fixed at 30. The number of flows is 20, and each flow has up to seven routing paths. From Figure 6.11, we see that increasing node degrees reduces the superframe length. A high node degree means a large number of links. Thus, the corresponding value of $\min\{c_{ij}\}$, which reduces the superframe length. In terms of computation time, see Figure 6.12, both the number of max cuts N and the number of links with non-zero load have an effect on computation time. The value of N may increase when there are more links in topology. This decides the number of decision variables in LP-CAPACITY, LP-MAIN and LP-NEWTM. Note that 20 flows and available paths for each flow are randomly regenerated in every simulation run. When node degrees increases, a flow may have one or more new routing paths with fewer hops, Thus, the number of idle links increases. As discussed in the second experiment, the number of links with non-zero load has a direct impact on the number of iterations carried out by LP-NEWTM and the number of constraints in LP-MAIN. This is particularly significant when the node degree ranges from 3 to 6. However, at node degree of 7, the number of max cuts, i.e., N, increases significantly, which causes

the computation time of Algo-PolyH to rise slightly.

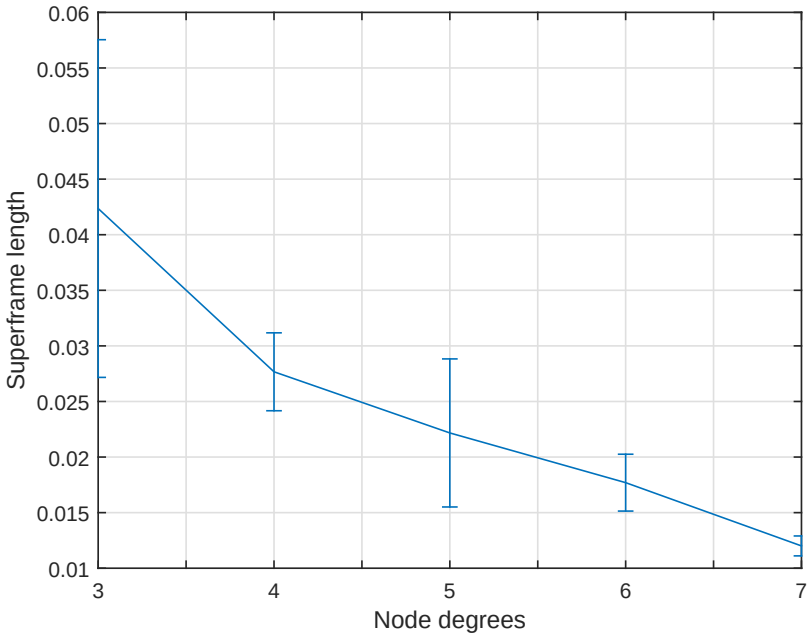


Figure 6.11: Average superframe length with different number of node degrees

As mentioned in Section 6.1, Algo-PolyH relies on Algo-2 [32] to heuristically derive the matrix **A**. Consequently, a question of interest is whether the number of max cuts has any impact on performance. A small topology is used in which generating all possible max cuts is computationally tractable. Specifically, this section will use the topology shown in Figure 6.13. It has five nodes and 16 links; dashed lines correspond to bidirectional links. This section computes two **A** matrices; each column is a transmission set or max cut where a '1' indicates a link is active. The first matrix, labeled as A1, contains all possible max cuts; i.e., A1 contains 65535 $(2^{16}-1)$ columns. The second matrix, called A2, contains 22 max cuts generated by Algo-2. Note, in A2, there are 16 columns; each of which contains one active link. Given a polyhedral set, the performance of Algo-PolyH is compared when using either A1 or A2. Specifically, this section records the resulting superframe length, computation time and the number of iterations executed by Algo-PolyH.

For the said topology, the resulting superframe length when using A1 and A2 is 0.0250 and 0.0667, respectively. However, the computation time when using A1 is

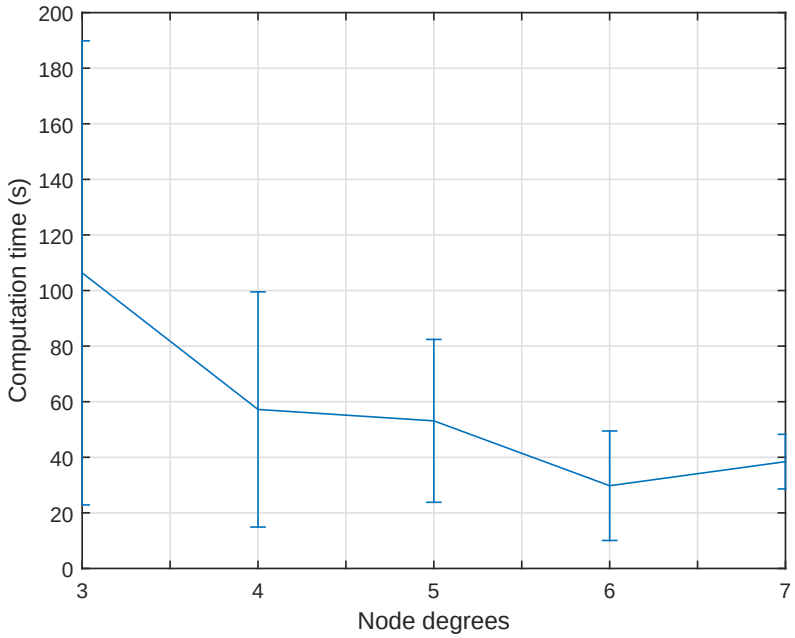


Figure 6.12: Average computation time with different number of node degrees

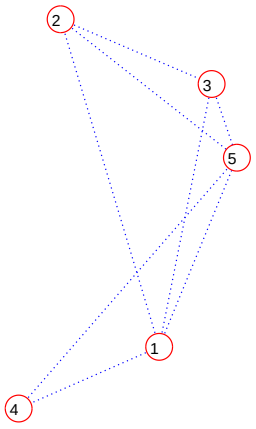


Figure 6.13: A small example topology

3.0356 seconds, significantly higher than 0.7875 second required for A2. For both cases, the number of iterations needed to find the final result is 2. These results show that Algo-PolyH can produce shorter superframes using all possible max-cuts. The downside, however, is the increasing computation time, which can be infeasible for larger sized networks. Recall that the number of columns in matrix A1 increases exponentially with the number of links, and the number of columns determines the number of decision variables in LP-MAIN; please see Proposition 12.

6.5 Conclusion

This chapter has studied a new problem that calls for a *robust* routing and superframe that support all TMs within a given polyhedral model. Advantageously, the derived solution remains feasible for all TMs within the model. The problem is challenging because a given polytope has an infinite number of TMs and a finite number of extreme points. The proposed solution, called Algo-PolyH, employs four different LPs to solve the problem iteratively to generate a robust solution. Section 6.4 has quantified the performance of Algo-PolyH in networks with different flows, paths per source, node degrees and number of nodes. The experiments confirm that Algo-PolyH is able to yield a robust solution as well as produce routings that yield minimal superframe lengths.

Up until this point, all the algorithms outlined in Chapter 3, 4, 5, and 6 do not consider nodes with MIMO capability. This is an important consideration because as mentioned in Chapter 1, MTR WMNs can be realized by equipping nodes with multiple antennas. Given a MIMO-based MTR WMN, a link scheduler must consider how interference is nulled/suppressed; i.e., how DoFs at the transmitter or/and receiver are allocated for data transmission or/and interference cancellation (IC). Therefore, the next chapter will present an efficient scheduler that allocates DoFs and generates the minimum superframe length for MIMO-based MTR WMNs.

Chapter 7

An Efficient Link Scheduler for MIMO-based MTR WMNs

In MIMO-based WMNs, the Degree of Freedoms (DoFs) or antenna elements available at each node enable concurrent transmission/reception of multiple independent data streams. Alternatively, these DoFs can be used to suppress interference. Recall that all interference streams have to be nulled/suppressed by the transmitter or/and receiver side. However, to date, all existing MTR schedulers do not consider the DoF assignment problem when deriving a link schedule. As discussed in Chapter 2, the authors of [29] proposed an efficient DoF model that ensures only one end of a link uses its DoFs for IC, but not both. Past works that have applied this model aim to maximize the minimum flow rate in a network [73] [75]. However, no works have studied how this model impacts the superframe length. Therefore, this chapter studies the link scheduling problem in MIMO-based MTR WMNs.

Given the traffic demand (weight) of each link, this chapter aims to design a scheduler that derives the shortest TDMA superframe that satisfies all link demands through joint stream control, node ordering and MTR. The problem at hand can be separated into two sub-problems: (1) ordering nodes to efficiently make use of their

DoFs, and (2) minimizing the superframe length by activating the maximal number of data streams in each slot.

This chapter outlines a novel solution or scheduler, called Algo-MIMO, that takes advantage of the node ordering DoF model of [29] and MTR capability afforded by MIMO technology to maximize the number of activated data streams in each slot. In turn, this helps minimize superframe length, and improves network capacity. Numerical results show that Algo-MIMO is able to reduce the superframe length by up to 60% as compared to algorithms that use other IC models, and approximately 40% against algorithms that order nodes based on the number of neighbors, node weight or randomly.

7.1 Preliminaries

Consider an arbitrary MIMO-based WMN $G(V, E)$, where V is the set of nodes, and E represents the set of links. Let (i, j) or $e_{ij} \in E$ indicate the link from node i to j, where $i, j \in V$. Assume time is divided into time slots. A superframe S is a grouping of slots. Thus, the superframe length $|S|$ is the number of slots. Each node i has MIMO capability and operates over the same frequency [36]. MIMO technology allows a node to transmit to or receive from multiple neighbors simultaneously via spatial multiplexing over one or more data streams. This MTR capability can be achieved using minimum mean square error sequential interference cancellation (MMSE-SIC), which allows multiple streams to be decoded at a receiver [25]. However, nodes must not transmit and receive at the same time; i.e., all links are half-duplex. This can be defined as the *no Mix-Tx-Rx* constraint.

Constraint 1. *A link schedule is feasible if all nodes satisfy the* no Mix-Tx-Rx *constraint.*

A node is able to transmit multiple data streams to one neighbor in an assigned slot. This is called *link upgrade* in this chapter. Consider a link $e \in E$ with weight

$w(e)$ that indicates the number of streams required in the final superframe. If link e can activate $w(e)$ streams in slot k simultaneously, then this link only needs to be assigned one slot. The maximum number of data streams on link (i, j) is bounded by $MIN(A_i, A_j)$, where A_i and A_j indicate the number of DoFs at node i and j respectively.

A node also needs a subset of DoFs to null/suppress interference caused to/from neighbors. Assume the interference range of a node is equal to its transmission range. A transmitter's data streams that interfere at a neighbor or un-intended receiver are called *interfering streams* from the said receiver's perspective. The receiver will need the same number of DoFs to suppress these interfering streams. Alternatively, the transmitter can null its transmission toward the receiver. The number of DoFs to null interfering streams on the transmitter side depends on the total number of data streams at interfered receivers. As shown in Figure 7.1, node A is transmitting one data stream to node B, and node C is transmitting two data streams to node D. Node C has two interfering streams to node B. If node B wants to suppress the interference streams from node C, node B must assign two DoFs for IC. If node C wants to null the interference to node B, it must assign one DoF for IC because node A has one data stream to node B.

As mentioned in [30] and [29], either the transmitter (node C) or the receiver (node B) needs to assign DoFs for IC. Requiring both sides to perform IC will waste DoFs [73]. This chapter adopts the DoF model proposed in [29] to ensure all interference streams are canceled by one end of a link only. Specifically, given a sorted list of nodes, to perform IC, DoFs are assigned based on the position of a node in the given list. A transmitter is responsible for nulling its transmission to unintended receivers that are ordered before it; a receiver needs to suppress the interfering streams from unintended transmitters that are ordered before it. Let $D_i^+ \leq A_i$ denote the DoFs used by node i to transmit, and I_i^+ refers to DoFs used to null its interference. Also $D_j^- \leq A_j$ denotes the DoFs used by node j to receive, and I_j^- for DoFs used to suppress interfering streams. Let o_i and o_j indicate the

order of node i and j respectively. Then, the *node ordering IC* constraint is,

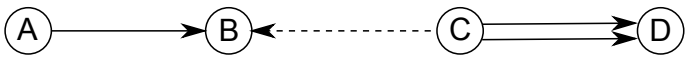


Figure 7.1: An interference example

Constraint 2. *Each interfering stream must be cancelled by the end with the bigger node order; i.e., if node i interferes with j and $o_i > o_j$, then node i assigns I_i^+ DoFs for IC, where $I_i^+ \geq D_j^-$.*

The *DoF* constraint is shown below.

Constraint 3. *The total number of elements used for data transmission/reception and IC by node i must not exceed the total number of antenna elements: $D_i^+ + I_i^+ \leq A_i$ or $D_i^- + I_i^- \leq A_i$.*

According to Constraint 3, a transmitter/receiver i can use up to $A_i - 1$ DoFs for IC, with at least one remaining DoF dedicated for data stream. If a data stream is to be added, both ends of this stream must satisfy the *no Mix-Tx-Rx*, *node ordering IC* and *DoF* constraints.

The proposed algorithm is based on Algo-2, a centralized MTR link scheduler proposed in [32]. Algo-2 is able to generate a TDMA schedule that maximizes network capacity with a short superframe length. In each slot, it generates the maximum set of links that can be activated simultaneously, which is the well-known, NP-complete, MAXCUT problem [99]. Specifically, it creates the maximal bipartite graph in each slot and derives the shortest superframe length that ensures all link demands are satisfied. In slot k, Algo-2 divides nodes into two sets: Set1 and Set2. The nodes in Set1 will transmit to nodes in Set2 in slot k. The two sets are constructed as follows. Algo-2 first includes all nodes into Set2 and sets Set1 to empty. It then selects the node with the highest Δ value in Set2 and moves it into Set1, where the Δ value of a node is the difference between the total weight of links from this node to other nodes in Set2 and the sum of link weights from nodes in Set1 to this node. Algo-2 then repeatedly moves nodes from Set2 to Set1 until the Δ

value of all nodes in Set2 is less than or equal to zero. After determining the links in one slot, the weight of activated links is reduced by one. The algorithm terminates when the weight of all links is zero. A key problem with Algo-2 is that it assumes nodes always have sufficient number of antennas for IC. Without consideration for link upgrade and IC, Algo-2 is unable to generate the maximal number of streams in MTR WMNs. Section 7.2 will show how this is addressed.

7.2 Algorithm

This section outlines the details of Algo-MIMO. It generates a superframe S that satisfies all link demands (weights) while considering Constraint 1, 2 and 3. In each slot, Algo-MIMO operates using the following key ideas. It assigns ordering to nodes based on the two node sets generated by Algo-2. Then, it adds links between these two sets one by one provided nodes satisfy Constraint 3. Finally, Algo-MIMO checks each activated link to find whether it is able to support more data streams.

Algo-MIMO takes as input a graph $G(V, E)$, $\mathcal{A}$ and $\mathcal{W}$, where $\mathcal{A}$ contains the number of DoFs at each node, and $\mathcal{W}$ is the set of link weights. Let s_k indicate the k-th slot. The order number is represented by α. Let $\mathcal{L}$ be the set of links crossing Set1 into Set2 with non-zero weights. Let L_i represent the number of incident links on node i in set $\mathcal{L}$. In slot s_k, Algo-MIMO has three phases: (1) assign an order to nodes, (2) schedule links with one data stream, and (3) link upgrade. In *Phase-1*, it first applies Algo-2 [32] to separate nodes into two sets, Set1 and Set2; see *line 4* of Algorithm 4. Algo-MIMO generates the set $\mathcal{L}$ and computes L_i for each node. For each node set, it reorders nodes according to L_i in non-increasing order; see *line 5-10*. The function *AssignOrder()* allocates the node order α to the first unlabelled node in a node set; see *line 11-16*. For example, the first node in Set1 is labeled 1, and the first node in Set2 is labeled 2.

In *Phase-2*, Algo-MIMO checks each link from Set1 to Set2 with non-zero weight. It first adds one data stream to link (i, j) into slot s_k, where $i \in Set1$ and $j \in$

*Set*2. The function *CheckDoFs()* is used to check whether all nodes satisfy the node ordering IC constraint and DoFs constraint. If the result of *CheckDoFs()* is TRUE, Algo-MIMO will reduce the weight of link (i, j) by one. Otherwise, the link (i, j) will not be activated in slot s_k; see *Line 17-26*.

In *Phase-3*, Algo-MIMO adds data streams to be activated links in slot s_k. It continues adding streams to a link until its weight is zero or a node becomes overloaded; see *line 27-36*. Algo-MIMO repeats the three phases until all link weights are satisfied; see *line 2 and 37*.

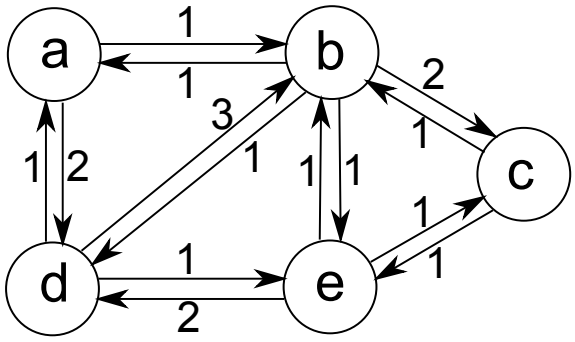


Figure 7.2: An example topology with link weights

Consider the following example. Specifically, it shows how Algo-MIMO generates a superframe for the example in Figure 7.2. Algo-MIMO takes as input $G(V, E)$, the set of DoFs $\mathcal{A} = \{3, 3, 3, 3, 3\}$ and the set of link weights $\mathcal{W}$. In slot s_1, it applies Algo-2 and generates two node sets, $Set1 = \{a, b, e\}$ and $Set2 = \{c, d\}$. The links from Set1 to Set2 are $\mathcal{L} = \{(a, d), (b, c), (b, d), (e, c), (e, d)\}$. Thus, the number of incident links of each node in Set1 are $\{L_a = 1, L_b = 2, L_e = 2\}$. In Set2, the values are $\{L_c = 2, L_d = 3\}$. Then, in each node set, it reorders nodes according to the number of incident links. The two node sets are $Set1 = (b, e, a)$ and $Set2 = (d, c)$. It labels the first node in Set1, i.e., b, with 1, and the first node in Set2, i.e., d, with 2. Then, the second node in Set1 is labeled 3. Thus, the node ordering in Set1 and Set2 are $\{o_b = 1, o_e = 3, o_a = 5\}$ and $\{o_d = 2, o_c = 4\}$, respectively. Algo-MIMO first adds link (b, d) into slot s_1. Node b uses one DoF for transmission. Node d uses one DoF for reception. All other nodes have unused DoFs. *CheckDoFs()* returns TRUE because all nodes satisfy the DoF constraint. Thus, the weight of link (b, d)

Algorithm 4: Algo-MIMO

input : $G(V, E)$, $\mathcal{A}$, $\mathcal{W}$
output: link schedule S

$k \leftarrow 0$
while $MAX(\mathcal{W}) > 0$ **do**
  $k \leftarrow k + 1$, $s_k \leftarrow \emptyset$, $\alpha \leftarrow 0$
  `// Phase 1: Sort and label nodes`
  $\{Set1, Set2\} \leftarrow$ `Algo-2(`G`)`
  $\mathcal{L} \leftarrow$ `RetrieveLinks(`*Set1,Set2*`)`
  for $i \leftarrow 1$ **to** $|V|$ **do**
    $L_i \leftarrow$ `IncidentLinks(`i,$\mathcal{L}$`)`
  end
  Set1$\leftarrow$ `ReorderNode(`*Set1*,L_i`)`
  Set2$\leftarrow$ `ReorderNode(`*Set2*,L_i`)`
  while $\alpha < |V|$ **do**
    $\alpha \leftarrow \alpha + 1$
    `AssignOrder(`α,*Set1*`)`
    $\alpha \leftarrow \alpha + 1$
    `AssignOrder(`α,*Set2*`)`
  end
  `// Phase 2: Add one data stream per link`
  for $i \leftarrow 1$ **to** $|\mathcal{L}|$ **do**
    if $w(e_i) > 0$ **then**
      Add one stream to link e_i in slot s_k
      if `CheckDoFs(`G,s_k,$\mathcal{A}$`)` **then**
        $w(e_i) \leftarrow w(e_i) - 1$
      else
        Delete link e_i from slot s_k
      end
    end
  end
  `// Phase 3: Link upgrade`
  for $n \leftarrow 1$ **to** $|s_k|$ **do**
    while $w(e_n) > 0$ **and** `CheckDoFs(`G,s_k,$\mathcal{A}$`)` **do**
      Add one more stream to link e_n in slot s_k
      if `CheckDoFs(`G,s_k,$\mathcal{A}$`)` **then**
        $w(e_n) \leftarrow w(e_n) - 1$
      else
        Delete one stream to link e_n from slot s_k
      end
    end
  end
  Add slot s_k into S
end
Return S

reduces by one. It then adds link (b, c) into slot s_1. The result of *CheckDoFs()* is TRUE because the total number of used DoFs is less than the number of elements at each activated node. Thus, the weight of link (b, c) reduces by one. Similarly, links (e, d), (e, c) and (a, d) are added into slot s_1 because no node is overloaded. Then, it checks each activated link in slot s_1. Link (b, d) cannot have more streams because its weight is zero. It adds one more data stream to link (b, c). Node b uses all its DoFs for transmission whereas node c uses all DoFs for reception. *CheckDoFs()* returns TRUE because no node is overloaded. Thus, the weight of link (b, c) reduces by one. It cannot add more streams to link (b, c) because its weight is zero. It then adds one more data stream to link (e, d). Node d has no unused DoFs for one more data stream. Thus, *CheckDoFs()* returns FALSE, and Algo-MIMO deletes the new added data stream to link (e, d). Similarly, it cannot add more data streams to link (a, d) and (e, c). Thus, in slot s_1, Algo-MIMO adds one more stream to link (b, c). The final superframe is shown in Table 7.1.

Slot 1	**Slot 2**	**Slot 3**	**Slot 4**	**Slot 5**
(b, d)	(d, b)	(d, b)	(a, d)	(b, a)
(b, c)	(d, e)	(d, a)	(b, e)	(e, d)
(b, c)	(c, b)	(e, b)		
(e, d)	(c, e)	(d, b)		
(e, c)	(a, b)			
(a, d)				

Table 7.1: Superframe generated by Algo-MIMO

This section ends with the run time complexity of Algo-MIMO.

Proposition 14. *Algo-MIMO has a running time complexity of*

$O(\frac{|V|^3}{4} \times MIN(MAX(\mathcal{W}), MAX(\mathcal{A})))$

Proof. The time complexity of line 1 and 3 is $O(1)$. According to [32], the time complexity of Algo-2 is $O(|V|^2)$. The maximum size of Set1 and Set2 is $\frac{|V|}{2}$. Thus, the time complexity of line 5 is $O(\frac{|V|^2}{4})$. Line 6-8 requires $O(|V|)$. Line 9 and 10 reorder nodes in each set, meaning they have a time complexity of $O(\frac{|V|}{2} log(\frac{|V|}{2}))$. For line 11-16, the time complexity is $O(|V|)$. Thus, the running time complexity of line 3-16 is $O(|V|^2)$. The number of iterations from line 17 to 26 is dependent on $|\mathcal{L}|$. The function *CheckDoFs()* needs to check a maximum $|V|$ nodes. Thus, the time

complexity of line 17-26 is $O(\frac{|V|^3}{4})$. The number of iterations from line 27 to 36 is less than or equal to $|\mathcal{L}|$. The maximum number of data streams on a link in each slot is c. The time complexity of line 27-36 is $O(\frac{|V|^3}{4} \times MIN(MAX(\mathcal{W}), MAX(\mathcal{A})))$, which is also the time complexity of Algo-MIMO. □

7.3 Evaluation

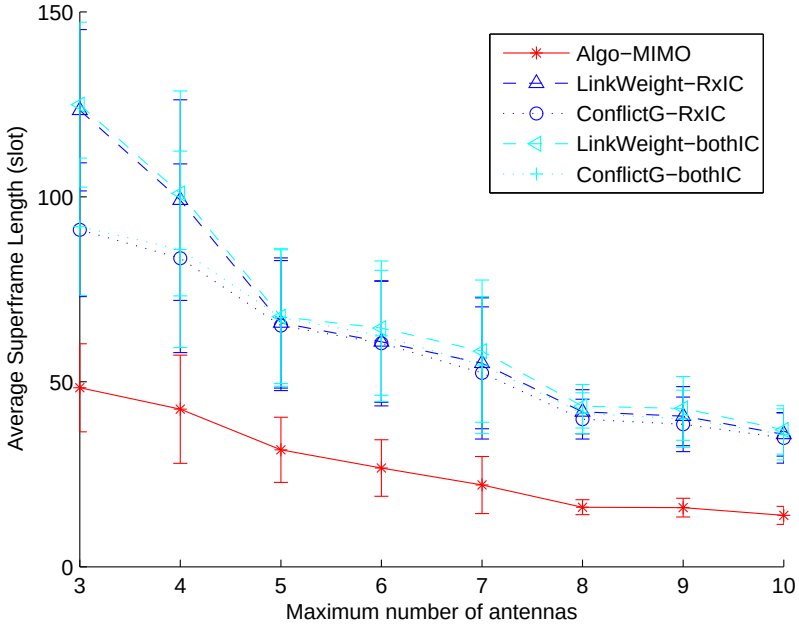
Algo-MIMO is validated using Matlab with the Matgraph toolkit [102]. All nodes are assumed stationary and located randomly on a 300 × 300 m^2 or 500 × 500 m^2 square area with a transmission range of 100 or 150 meters depending on the experiment. The number of nodes is 30 or 50. Assume that each node has a MIMO antenna array with the same number of DoFs that ranges from 3 to 10. In order to model different traffic load, each MIMO link is randomly allocated a weight between one to a maximum value t that corresponds to the number of data streams required by a link. The average superframe length is recorded out of 10 runs on different topologies. Channel state information (CSI) is assumed available at each node and nodes are located in a rich scattering environment. Assume each stream has the same data rate. Varying channel conditions is a consideration in an immediate future work.

There are two sets of experiments: without or with node ordering ordering rules. First, Algo-MIMO is evaluated against four algorithms without any node ordering rules. Namely, LinkWeight-RxIC, ConflictG-RxIC, LinkWeight-bothIC and ConflictG-bothIC. The labels 'RxIC' or 'bothIC' are used to indicate IC performed at the receiver side only or both ends of a link, respectively. LinkWeight-RxIC and LinkWeight-bothIC start with the highest weighted link. Both ConflictG-RxIC and ConflictG-bothIC first generate a contention graph where a directional edge from one vertex to another denotes a link interferes with another link. They then start with the link with the maximum in-degrees. LinkWeight-RxIC, ConflictG-RxIC, LinkWeight-bothIC and ConflictG-bothIC have the same link scheduling process

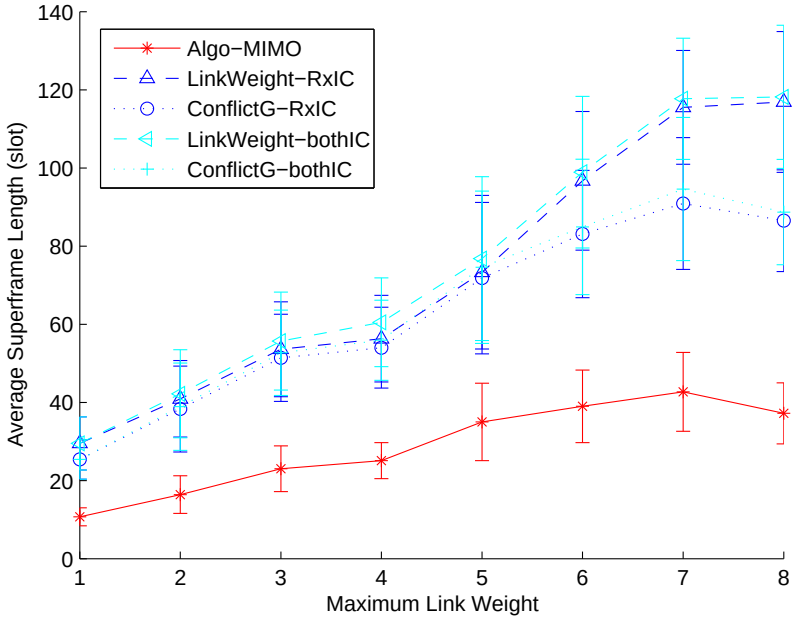
but use different IC rules. They assign a selected link to the first slot whereby all nodes satisfy the no Mix-Tx-Rx and DoF constraints.

In the second set of experiments, Algo-MIMO is compared against seven algorithms with different node ordering rules: Random, TxFirst, RxFirst, MaxNodeWeight, MinNodeWeight, MaxNeighbor and MinNeighbor. The difference between them and Algo-MIMO is how nodes are ordered. For all seven algorithms, Algo-2 [32] is first applied to derive Set1 and Set2. Recall that Set1 contains transmitters whilst those in Set2 are receivers. The Random method orders all nodes randomly. TxFirst first randomly labels all transmitting nodes in Set1 followed by Set2. Conversely, RxFirst starts with Set2 first. A node's weight is defined as the sum of the highest incoming and outgoing link weight. MaxNodeWeight first labels the node with the highest node weight. MinNodeWeight starts labelling from the link with the smallest node weight. MaxNeighbor and MinNeighbor start with the node that has the maximum and minimum number of neighbors respectively. After each node is labelled with an order number, the seven algorithms use the same *Phase-2* and *Phase-3* as Algo-MIMO.

Algo-MIMO is first evaluated against four algorithms without node ordering. It studies the effect of the number of DoFs on a 300×300 m^2 area with 30 nodes. It randomly assigns a weight to each link that ranges from 1 to 5. From Figure 7.3a, we see that Algo-MIMO generates the shortest superframe length when the DoFs range from 3 to 10. For all five algorithms, the average superframe length reduces with increasing DoFs. This is because more DoFs are available to support data streams in each slot. In low DOFs scenarios, LinkWeight-bothIC generates the longest superframe length, i.e., 124.9 slots. This is 2.58 times the superframe length generated by Algo-MIMO. ConflictG-RxIC and ConflictG-bothIC respectively derive superframes with an average length of 91.1 and 91.9 slots, respectively. This is about 90% longer than that of Algo-MIMO. When nodes have many DOFs, the superframe lengths generated by LinkWeight-RxIC, LinkWeight-bothIC, ConflictG-RxIC and ConflictG-bothIC are more than 150% longer than that of Algo-MIMO.



(a)



(b)

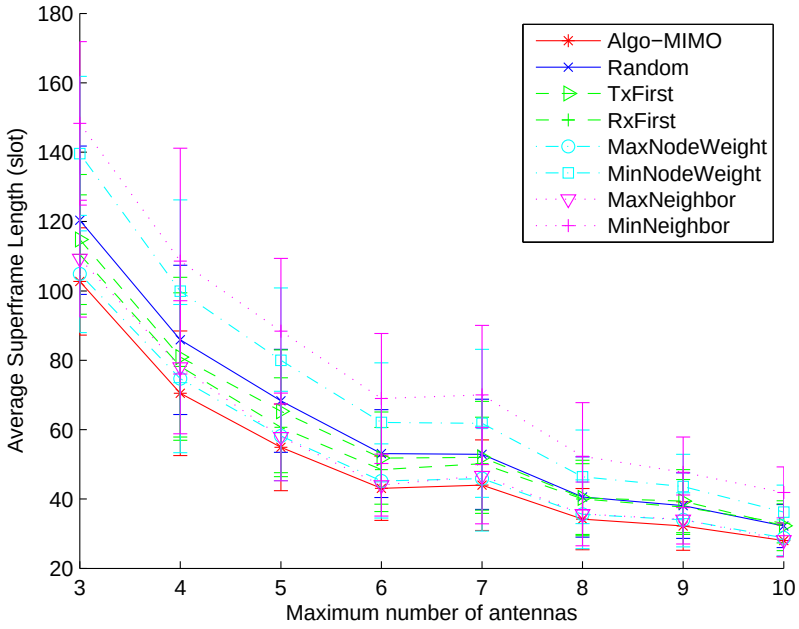
Figure 7.3: Comparison against algorithms without node ordering

Next, the effect of link weights is studied. Assume each node has five DoFs. Each link has a random weight between one and t, where t ranges from 1 to 8. From Figure 7.3b, we see that the superframe length increases proportionally with the value of t. A link may need more time slots to satisfy the higher link demand. When the maximum link weight is one, the superframe of LinkWeight-bothIC is about three times the result of Algo-MIMO. When the maximum link weight is 8, the superframe length of Algo-MIMO is only 30% of the superframe length generated by LinkWeight-bothIC.

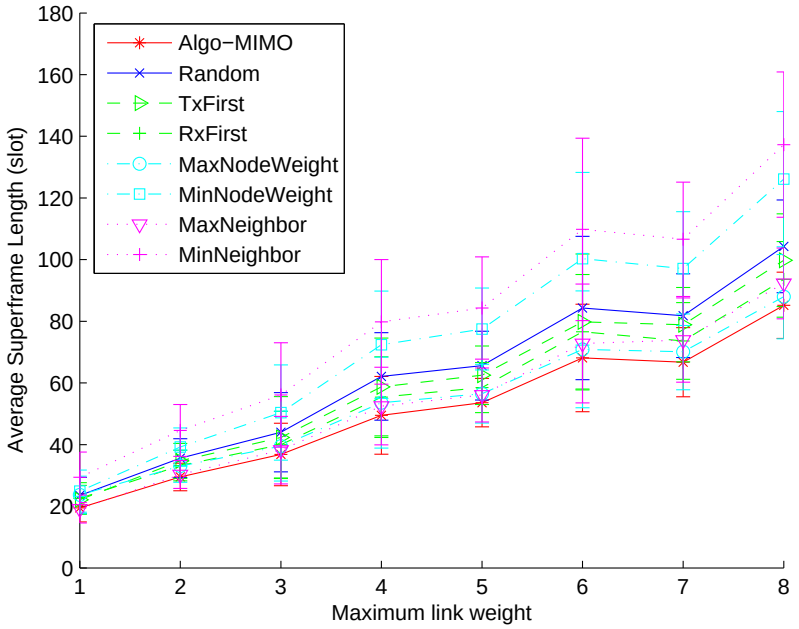
Algo-MIMO performs better because it applies the node ordering DoF model that cancels interference only on one end of a link. Consequently, as there are no wastage of DoFs, Algo-MIMO is able to activate more streams in each slot. The second reason is because Algo-MIMO uses Algo-2, which ensures the maximal number of links are activated in each slot. In contrast, LinkWeight-RxIC, LinkWeight-bothIC, ConflictG-RxIC and ConflictG-bothIC only generate feasible schedules without considering superframe length.

The second set of experiments evaluates the performance of Algo-MIMO against seven algorithms with different node ordering rules. It first studies the effect of DoFs at each node on the superframe length on a 500×500 m^2 area with 50 nodes when transmission range is 150 meters. It randomly assigns a weight to each link that ranges from 1 to 5. From Figure 7.4a, we see that the average superframe length reduces when there are more DoFs at each node. When each node has three DoFs, MinNeighbor has the longest superframe length, i.e., 148.3 slots. This superframe length is 44% longer than the result derived by Algo-MIMO. When the number of DoFs is 10, MinNeighbor derives the longest superframe, i.e., 41.9 slots, among the eight algorithms. This value is about 1.5 times the superframe length of Algo-MIMO.

The next experiment studies the effect of link weight on the superframe length. The maximum link weight t ranges from 1 to 8. From Figure 7.4b, we see that Algo-MIMO generates the shortest superframe in all cases. When the maximum link weight is one, MinNeighbor derives the longest superframe length, i.e., 29.4 slots.



(a)



(b)

Figure 7.4: Comparison against algorithms with different node ordering rules

This value is 1.5 times the superframe length of Algo-MIMO. When the maximum link weight is 8, the superframe length of MinNeighbor is about 1.6 times that of Algo-MIMO.

Algo-MIMO has better performance because it orders nodes according to the number of incident links in $\mathcal{L}$. Thus, in each slot, a node that has a high number of incident links will have a smaller label or node ordering. A node with a smaller label requires fewer DoFs for IC. Consequently, Algo-MIMO can activate more data streams in each slot than other ordering rules due to the efficient use of DoFs.

7.4 Conclusion

This chapter has presented the first novel algorithm that generates a schedule using an efficient node ordering rule for IC. The algorithm, called Algo-MIMO, is significant for MIMO-based MTR WMNs whereby nodes must decide how their DoFs are allocated to facilitate data transmissions or to null interferences. Experiment results show that the node ordering rule proposed in [29] leads to shorter superframe lengths as compared with rules such as random, transmitter or receiver first, node weight or number of neighbors.

Chapter 8

Conclusion

This thesis has investigated numerous link scheduling approaches for Wireless Mesh Networks (WMNs). The key aim is to support a large amount of traffic or network capacity. As shown in this thesis, these link schedulers have a direct impact on network capacity, and hence, the number of flows or traffic supported by a WMN. Unlike existing works, this thesis is set in the context of WMNs where nodes are capable of multiple transmissions or receptions, aka MTR, at the same time. Such MTR WMNs have much higher capacity than those that use omni or directional antennas. Moreover, they have a unique interference model and thus require novel link schedulers. Thus, the overarching problem addressed in this thesis is designing link schedulers that exploit the MTR capability of nodes to yield a schedule that meets one or more objectives.

To this end, this thesis makes several contributions. It contains a number of novel Time Division Multiple Access (TDMA) link schedulers. First, it studies the problem of optimizing both schedule length and end-to-end delays. This is a key concern to multimedia applications that require some guarantees on the delays experienced by flows. Second, it studies the problem of ensuring flows have a fair share of the network capacity. This is important because a derived schedule may starve flows. Lastly, this thesis takes the first step towards deriving a schedule that

is suitable for random demands. This is significant because demands are known to be uncertain or can be characterized by a probability distribution. Hence, any derived schedule must be robust against any changes. Otherwise, a central controller will be required to update the schedule frequently, which incurs significant signaling overheads.

In summary, this thesis has addressed the following research questions/problems:

1. Given a set of end-to-end traffic demands in an MTR WMN, derive a schedule that minimizes the end-to-end delay of these demands. This problem is novel and has not been considered in the context of MTR WMNs. Chapter 3 outlines an approach that simultaneously considers routing, construction of a minimal link schedule and reordering of transmission slots. In addition, it presents a Non-Linear Integer Program (NILP); unfortunately, the NILP is intractable due to the exponential number of possible routing paths and link scheduling problem that is equal to solving the NP-complete MAX CUT problem in each slot. Thus, Chapter 3 outlines two novel heuristic algorithms: JRS-Multi-DEC and JRS-BIP. The key idea is to first find the optimal routing that minimizes the maximum link load so that the resultant superframe length is minimized. Then, a novel slot re-ordering algorithm is used to further reduce end-to-end delays. As shown by the numerical results, both algorithms are able to reduce the superframe length by more than 45% and 70% as compared to JRS-Shortest and NJR, respectively. The end-to-end delay of flows is reduced by more than 50% as compared to NJR, and about 30% as compared to JRS-Shortest.

2. Another significant problem addressed in this thesis is scheduling links such that the resulting flow rates satisfy max-min fairness. In Chapter 4, a link scheduler, named Algo-Fair, is shown to directly generate a superframe that ensures flows have a fair rate. This scheduler is distinct from other schedulers that first use a conflict graph or a linear program to first compute a fair

allocation before deriving a schedule. Moreover, Algo-Fair is the first approach that derives a link schedule that ensures flows have a fair rate in MTR WMNs. Critically, it incorporates a general and novel augmentation step to distribute spare capacity fairly among flows. Numerical results show that the proposed algorithm can achieve the optimal MMF allocation in some cases. On average, the gap to the optimal solution is approximately 3.7%. Moreover, as compared with other designs, Algo-Fair is able to provide a higher minimum flow rate and average throughput.

3. Given random traffic loads, a fundamental problem is how to size the TDMA and random access parts of a hybrid superframe such that the resultant idle time and probability of collision are minimized in MTR WMNs. To this end, Chapter 5 presents the first two-stage stochastic program (SP) approach to dimension such a superframe. This is significant because in existing wireless networks, the boundary is usually predefined. On the other hand, in the proposed SP approach, the boundary between the scheduled and random access part is controlled by a penalty value. The optimal penalty value that ensures the probability of collision is less than a given threshold can then be determined using binary search. With the SP approach in hand, a central controller does not need to regenerate a superframe whenever traffic demands change. Hence, the proposed SP approach is suitable for large-scale WMNs.

4. A key gap in works that aim to optimize both routing and scheduling is consideration for uncertain traffic demands. To this end, this thesis contains the first JRS work that assumes traffic demands are characterized by a polyhedral model. Specifically, Chapter 6 outlines a novel JRS problem that aims to determine the routing and superframe that supports all extreme points in a given polytope. The problem is formalized as a semi-infinite LP. Then, a heuristic algorithm, named Algo-PolyH, is proposed to solve the LP. Algo-PolyH uses four LPs to solve the problem. Namely, LP-CAPACITY, LP-INIT-TM, LP-MAIN

and LP-NEWTM. LP-CAPACITY is used to find the maximum capacity of each link. Using the generated link capacity, LP-INIT-TM is applied to find a feasible traffic matrix (TM). Then, LP-MAIN generates a routing and schedule to support all found TMs. Lastly, LP-NEWTM checks whether there is an unsupported TM in the given polytope. If so, it finds a new routing and schedule that supports the TM found by LP-NEWTM. The robustness of the solution found by Algo-PolyH has been verified in networks with different number of flows, available paths per source, node degrees and number of nodes.

5. Recent advances in MIMO technologies have paved the way for high-capacity MIMO-based MTR WMNs. A key problem is to allocate the antenna elements or Degree of Freedoms (DoFs) at each node efficiently. That is, how to assign DoFs for data streams and interference cancellation in order to yield the minimum superframe length? In this respect, Chapter 7 presents a study on different DoFs assignment methods and contains a novel approach called Algo-MIMO. Its key idea is to apply a recently developed novel node ordering rule [29] to efficiently assign DoFs. Numerical results show that Algo-MIMO reduces the superframe length by 60% as compared to other algorithms that cancel interference at both transmitter and receiver sides. Compared with algorithms that order nodes based on their weight, the number of neighbors or randomly, Algo-MIMO is able to reduce the superframe by approximately 40%.

There are many possible directions for future research. For example, the problems addressed in Chapter 3 and 7 can be considered jointly. The resulting problem is how to route flows, construct the minimal superframe, reorder slots and allocate DoFs to minimize the total end-to-end delay in MIMO-based MTR WMNs. In addition, the problem of designing a distributed JRS approach that minimizes delays remains open for MTR WMNs. A key assumption in Chapter 4 is that routing paths for flows are given. Thus, another possible direction is to jointly consider

routing and scheduling to find the lexicographically max-min fair rate allocation. In Chapter 5, the two-stage SP algorithm is general and can be readily applied to other non-MTR WMNs. Thus, an immediate future work is to evaluate its efficacy in other wireless networks that employ a different interference model.

Bibliography

[1] I. F. Akyildiz, X. Wang, and W. Wang, "Wireless mesh networks: a survey," *Computer networks*, vol. 47, no. 4, pp. 445–487, 2005.

[2] A. D. Gore and A. Karandikar, "Link scheduling algorithms for wireless mesh networks," *IEEE Communications Surveys and Tutorials*, vol. 13, no. 2, pp. 258–273, 2011.

[3] M. Kas, B. Yargicoglu, I. Korpeoglu, and E. Karasan, "A survey on scheduling in ieee 802.16 mesh mode," *IEEE Communications Surveys and Tutorials*, vol. 12, no. 2, pp. 205–221, 2010.

[4] L. Akyildiz and X. Wang, *Wireless Mesh Networks*. Wiley, 1 ed., April 2009.

[5] E. C. L. Chan and G. Baciu, *Introduction to Wireless Localization: With iPhone SDK Examples*. Wiley-IEEE Press, 2012.

[6] Y. Wang, Z. Li, and W. Zhang, "A fully distributed p2p communications architecture for network virtual environments," in *International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, (Wuhan, China), Sept. 2011.

[7] A. A. Pirzada, M. Portmann, and J. Indulska, "Evaluation of multi-radio extensions to aodv for wireless mesh networks," in *ACM International Work-*

shop on Mobility Management and Wireless Access (MobiWac), (Terromolinos, Spain), pp. 45–51, Oct. 2006.

[8] J. Robinson, *Deployment and Assessment of Wireless Mesh Networks*. PhD thesis, Rice University, 2009.

[9] R. Flickenger, S. O. E. Pietrosemoli, M. Zennaro, and C. Fonda, "Very long distance wi-fi networks," in *Proceedings of the second ACM SIGCOMM workshop on Networked systems for developing regions (NSDR)*, (Seattle, WA, USA), Aug. 2008.

[10] D. Aguayo, J. Bicket, S. Biswas, R. Morris, B. Chambers, and D. De Couto, "MIT roofnet," in *International Conference on Mobile Computing and Networking (MobiCom)*, (San Diego, CA, USA), Sept. 2003.

[11] V. Gabale, B. Raman, K. Chebrolu, and P. Kulkarni, "Lokvaani: Demonstrating interactive voice in lo3," in *Proceedings of the ACM SIGCOMM*, (New Delhi, India), pp. 461–462, Aug. 2010.

[12] C.-H. Yu, D.-L. Yang, and A.-C. Liu, "A study of wimax implementation at university campuses," *International Journal of Future Generation Communication and Networking*, vol. 22, pp. 1–8, 2008.

[13] "Aruba networks, an HP company." http://www.arubanetworks.com/resources/.

[14] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "IEEE 802.11s: The WLAN mesh standard," *IEEE Wireless Communications*, vol. 17, no. 1, pp. 104–111, 2010.

[15] Y. Li, Y. Yang, and C. Cao, "A novel routing algorithm in distributed IEEE 802.16 mesh networks," *IEEE Communications Letters*, vol. 13, no. 10, pp. 761–763, 2009.

[16] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer, "Wildnet: Design and implementation of high performance wifi based long distance networks," in *4th USENIX Symposium on Networked Systems Design & Implementation*, (Cambridge, MA, USA), Apr. 2007.

[17] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, 2000.

[18] L. Badia, A. Erta, L. Lenzini, and M. Zorzi, "A general interference-aware framework for joint routing and link scheduling in wireless mesh networks," *IEEE Network*, vol. 22, no. 1, pp. 32–38, 2008.

[19] U. Kumar, H. Gupta, and S. R. Das, "A topology control approach to using directional antennas in wireless mesh networks," in *IEEE International Conference on Communications (ICC)*, (Istanbul, Turkey), June 2006.

[20] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vaidya, "Using directional antennas for medium access control in ad hoc networks," in *Proceedings of the 8th annual international conference on Mobile computing and networking (MobiCom)*, (Atlanta, Georgia, USA), pp. 59–70, Sept. 2002.

[21] B. Raman and K. Chebrolu, "Design and evaluation of a new MAC protocol for long-distance 802.11 mesh networks," in *Proceedings of the 11th annual international conference on Mobile computing and networking (MobiCom)*, (Cologne, Germany), Aug. 2005.

[22] R. Mudumbai, S. Singh, and U. Madhow, "Medium access control for 60 GHz outdoor mesh networks with highly directional links," in *28th IEEE International Conference on Computer Communications (INFOCOM)*, (Rio de Janeiro, Brazil), Apr. 2009.

[23] D. Gesbert, M. Shafi, D. shan Shiu, P. J. Smith, and A. Naguib, "From theory to practice: an overview of MIMO space-time coded wireless systems," *IEEE*

Journal on Selected Areas in Communications, vol. 21, no. 3, pp. 281–302, 2003.

[24] K. Sundaresan and M. A. Ingram, "Medium access control in ad hoc networks with mimo links: Optimization considerations and algorithms," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 350–365, 2004.

[25] S. Chu and X. Wang, "Opportunistic and cooperative spatial multiplexing in MIMO ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 18, no. 5, pp. 1610–1623, 2010.

[26] V. Gabale, B. Raman, P. Dutta, and S. Kalyanraman, "A classification framework for scheduling algorithms in wireless mesh networks," *IEEE Communications Surveys and Tutorials*, vol. 15, no. 1, pp. 199–222, 2013.

[27] M. Yazdanpanah, C. Assi, and Y. Shayan, "Optimal joint routing and scheduling in wireless mesh networks with smart antennas," in *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, (Montreal, QC, Canada), June 2010.

[28] 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 (ICC)*, (Beijing, China), May 2008.

[29] Y. Shi, J. Liu, C. Jiang, C. Gao, and Y. Hou, "An optimal link layer model for multi-hop MIMO networks," in *30th IEEE International Conference on Computer Communications (INFOCOM)*, (Shanghai, China), Apr. 2011.

[30] D. M. Blough, G. Resta, P. Santi, R. Srinivasan, and L. M. Cortes-Pena, "Optimal one-shot scheduling for MIMO networks," in *8th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, (Salt Lake City, UT, USA), June 2011.

[31] M. Cheng, Q. Ye, and L. Cai, "Cross-layer schemes for reducing delay in multihop wireless networks," *IEEE Transactions on Wireless Communications*, vol. 12, no. 2, pp. 928–937, 2013.

[32] H. Loo, S. Soh, and K.-W. Chin, "On improving capacity and delay in multi Tx/Rx wireless mesh networks with weighted links," in *19th Asia-Pacific Conference on Communications (APCC)*, (Bali, Indonesia), Aug. 2013.

[33] P. Djukic and S. Valaee, "Delay aware link scheduling for multi-hop tdma wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, no. 3, pp. 870–883, 2009.

[34] C. W. Pyo and H. Harada, "Throughput analysis and improvement of hybrid multiple access in ieee 802.15.3c mm-wave WPAN," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 8, pp. 1414–1424, 2009.

[35] X. Meng, S. H. Wong, Y. Yuan, and S. Lu, "Characterizing flows in large scale wireless data networks," in *Proceedings of the 10th annual international conference on Mobile computing and networking (MobiCom)*, (Philadelphia, PA, USA), Sept. 2004.

[36] K. Sundaresan, R. Sivakumar, and M. A. Ingram, "A fair medium access control protocol for ad-hoc networks with MIMO links," in *The 23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, (Hong Kong, China), Mar. 2004.

[37] M. Kodialam and T. Nandagopal, "Characterizing achievable rates in multihop wireless networks: The joint routing and scheduling problem," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking (MobiCom)*, (San Diego,CA, USA), pp. 42–54, Sept. 2003.

[38] V. Friderikos and K. Papadaki, "Interference aware routing for minimum frame

length schedules in wireless mesh networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2008, no. 817876, pp. 1–13, 2008.

[39] B. Wang, M. Mutka, and E. Torng, "Optimization based rate allocation and scheduling in TDMA based wireless mesh networks," in *Proceedings of the 16th annual IEEE International Conference on Network Protocols (ICNP)*, (Orlando, Florida, USA), pp. 147–156, Oct. 2008.

[40] F. P. Kelly, A. K. Maulloo, and D. K. Tan, "Rate control for communication networks: shadow prices, proportional fairness and stability," *Journal of the Operational Research society*, vol. 49, no. 3, pp. 237–252, 1998.

[41] F. Gavril, "Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph," *SIAM Journal on Computing*, vol. 1, no. 2, pp. 180–187, 1972.

[42] P. Heggernes, "Minimal triangulations of graphs: A survey," *Discrete Mathematics*, vol. 306, no. 3, pp. 297–317, 2006.

[43] P. Serafini and W. Ukovich, "A mathematical model for periodic scheduling problems," *SIAM Journal on Discrete Mathematics*, vol. 2, no. 4, pp. 550–581, 1989.

[44] P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini, "Optimal joint routing and link scheduling for real-time traffic in TDMA wireless mesh networks," *Computer Networks*, vol. 57, no. 11, pp. 2301–2312, 2013.

[45] B. Gendron, A. Hertz, and P. St-Louis, "On edge orienting methods for graph coloring," *Journal of Combinatorial Optimization*, vol. 13, no. 2, pp. 163–178, 2007.

[46] L. Badia, A. Botta, and L. Lenzini, "A genetic approach to joint routing and link scheduling for wireless mesh networks," *Ad Hoc Networks*, vol. 7, no. 4, pp. 654–664, 2009.

[47] Z. Michalewicz, *Genetic algorithms+ data structures= evolution programs*. springer, 1998.

[48] B. Wang, G. Zeng, M. Mutka, and L. Xiao, "Routing for minimum length schedule in multi-channel TDMA based wireless mesh networks," in *11th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, (Montreal, QC, Canada), pp. 1–6, June 2010.

[49] J. Zhang, H. Wu, Q. Zhang, and B. Li, "Joint routing and scheduling in multi-radio multi-channel multi-hop wireless networks," in *2nd International Conference on Broadband Networks (BROADNETS)*, (Boston, MA, USA), pp. 631–640, Oct. 2005.

[50] A. Mehrotra and M. A. Trick, "A column generation approach for graph coloring," *informs Journal on Computing*, vol. 8, no. 4, pp. 344–354, 1996.

[51] J. El-Najjar, C. Assi, and B. Jaumard, "Joint routing and scheduling in wimax-based mesh networks," *IEEE Transactions on Wireless Communications*, vol. 9, no. 7, pp. 2371–2381, 2010.

[52] A. Capone, G. Carello, I. Filippini, S. Gualandi, and F. Malucelli, "Routing, scheduling and channel assignment in wireless mesh networks: Optimization models and algorithms," *Ad Hoc Networks*, vol. 8, no. 6, pp. 545–563, 2010.

[53] V. Gabale, A. Chiplunkar, B. Raman, and P. Dutta, "Delaycheck: Scheduling voice over multi-hop multi-channel wireless mesh networks," in *Third International Conference on Communication Systems and Networks (COMSNETS)*, (Bangalore, India), pp. 1–10, Jan. 2011.

[54] H. Shetiya and V. Sharma, "Algorithms for routing and centralized scheduling to provide QoS in IEEE 802.16 mesh networks," in *Proceedings of the 1st ACM Workshop on Wireless Multimedia Networking and Performance Modeling (WMuNeP)*, (Montreal, Quebec, Canada), pp. 140–149, Oct. 2005.

[55] W.-H. Tam and Y.-C. Tseng, "Joint multi-channel link layer and multi-path routing design for wireless mesh networks," in *26th IEEE International Conference on Computer Communications (INFOCOM)*, (Anchorage, Alaska, USA), pp. 2081–2089, May 2007.

[56] P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini, "Link scheduling with end-to-end delay constraints in wireless mesh networks," in *10th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, (Kos Island, Greece), pp. 1–9, June 2009.

[57] H. Lu, S. Liu, and A. 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 (WASA)*, (Chicago, USA), pp. 55–62, Aug. 2007.

[58] M. S. Kodialam and T. Nandagopal, "Characterizing achievable rates in multi-hop wireless mesh networks with orthogonal channels," *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 868–880, 2005.

[59] P. Dutta, S. Jaiswal, D. Panigrahi, and R. Rastogi, "A new channel assignment mechanism for rural wireless mesh networks," in *27th IEEE International Conference on Computer Communications (INFOCOM)*, (Phoenix, AZ, USA), pp. 2261–2269, Apr. 2008.

[60] P. Dutta, V. Mhatre, D. Panigrahi, and R. Rastogi, "Joint routing and scheduling in multi-hop wireless networks with directional antennas," in *29th IEEE International Conference on Computer Communications (INFOCOM)*, (San Diego, CA, USA), pp. 1–5, Mar. 2010.

[61] H. Su and X. Zhang, "Joint link scheduling and routing for directional-antenna based 60 GHz wireless mesh networks," in *Proceedings of the Global Communications Conference (GLOBECOM)*, (Honolulu, Hawaii, USA), pp. 1–6, Nov. 2009.

[62] 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.

[63] J. Crichigno, M.-Y. Wu, S. K. Jayaweera, and W. Shu, "Throughput optimization in multihop wireless networks with multipacket reception and directional antennas," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 7, pp. 1206–1213, 2011.

[64] J. Mietzner, R. Schober, L. Lampe, W. H. Gerstacker, and P. A. Hoeher, "Multiple-antenna techniques for wireless communications-a comprehensive literature survey," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 2, pp. 87–105, 2009.

[65] R. Bhatia and L. Li, "Throughput optimization of wireless mesh networks with MIMO links," in *26th IEEE International Conference on Computer Communications (INFOCOM)*, (Anchorage, Alaska, USA), pp. 2326–2330, May 2007.

[66] J.-S. Liu, C.-H. R. Lin, and K.-Y. Tung, "Cross-layer design for end-to-end throughput maximization and fairness in MIMO multihop wireless networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, no. 37, pp. 1–14, 2010.

[67] Y. Xu, S. Wan, J. Tang, and R. S. Wolff, "Interference aware routing and scheduling in wireless backhaul networks with smart antennas," in *Proceedings of the 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, (Rome, Italy), pp. 1–9, June 2009.

[68] M. Cao, X. Wang, S.-J. Kim, and M. Madihian, "Multi-hop wireless backhaul networks: A cross-layer design paradigm," *IEEE Journal on Selected Areas in Communications*, vol. 25, no. 4, pp. 738–748, 2007.

[69] R. A. Iltis, S.-J. Kim, and D. A. Hoang, "Noncooperative iterative mmse beamforming algorithms for ad hoc networks," *IEEE Transactions on Communications*, vol. 54, no. 4, pp. 748–759, 2006.

[70] M. Cheng and Q. Ye, "A combinatorial solution for scheduling spatial multiplexing in MIMO-based ad hoc networks," in *IEEE GLOBECOM Workshops*, (Anaheim, California, USA), Dec. 2012.

[71] K. Sundaresan, W. Wang, and S. Eidenbenz, "Algorithmic aspects of communication in ad-hoc networks with smart antennas," in *Proceedings of the 7th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*, (Florence, Italy), May 2006.

[72] D. Guo, Y. He, Y. Liu, P. Yang, X.-Y. Li, and X. Wang, "Link scheduling for exploiting spatial reuse in multihop MIMO networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1355–1365, 2013.

[73] X. Qin, X. Yuan, Y. Shi, Y. T. Hou, W. Lou, and S. F. Midkiff, "On throughput maximization for a multi-hop MIMO network," in *IEEE 10th International Conference on Mobile Ad-Hoc and Sensor Systems (MASS)*, (Hangzhou, China), Oct. 2013.

[74] Y. T. Hou, Y. Shi, and H. D. Sherali, *Applied optimization methods for wireless networks*. Cambridge University Press, 2014.

[75] H. Zeng, Y. Shi, Y. T. Hou, and W. Lou, "An efficient DoF scheduling algorithm for multi-hop MIMO networks," in *IEEE International Conference on Computer Communications (INFOCOM)*, (Turin, Italy), Apr. 2013.

[76] 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 (ICC)*, (Beijing, China), May 2008.

[77] B. Mumey, J. Tang, and T. Hahn, "Algorithmic aspects of communications in multihop wireless networks with MIMO links," in *IEEE International Conference on Communications (ICC)*, (Cape Town, South Africa), May 2010.

[78] D. Bertsekas and R. Gallager, *Data Networks*. Prentice-Hall, 1992.

[79] L. Tassiulas and S. Sarkar, "Maxmin fair scheduling in wireless ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 163–173, 2005.

[80] S. Sarkar and L. Tassiulas, "End-to-end bandwidth guarantees through fair local spectrum share in wireless ad-hoc networks," *IEEE Transactions on Automatic Control*, vol. 50, no. 9, pp. 1246–1259, 2005.

[81] T. Salonidis and L. Tassiulas, "Distributed on-line schedule adaptation for balanced slot allocation in wireless ad hoc networks," in *12th IEEE International Workshop on Quality of Service*, (Montreal, Canada), June 2004.

[82] A. Penttinen, I. Koutsopoulos, and L. Tassiulas, "Low-complexity distributed fair scheduling for wireless multi-hop networks," in *First Workshop on Resource Allocation in Wireless Networks*, (Italy), Apr. 2005.

[83] M. Pioro, M. Zotkiewicz, B. Staehle, D. Staehle, and D. Yuan, "On max-min fair flow optimization in wireless mesh network-," *Ad Hoc Networks*, vol. 13, pp. 134–152, 2014.

[84] P. Wang, H. Jiang, W. Zhuang, and H. V. Poor, "Redefinition of max-min fairness in multi-hop wireless networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 12, pp. 4786–4791, 2008.

[85] H. Wang and W. Jia, "Design a novel fairness model in wimax mesh networks," *Computer Communications*, vol. 35, no. 12, pp. 1447–1456, 2012.

[86] P. Tune and M. Roughan, "Internet traffic matrices: A primer," *Recent Advances in Networking*, vol. 1, 2003.

[87] W. Ben-Ameur and H. Kerivin, "Routing of uncertain traffic demands," *Optimization and Engineering*, vol. 6, no. 3, pp. 283–313, 2005.

[88] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. K. Ramakrishnan, and J. E. van der Merive, "A flexible model for resource management in virtual private networks," in *Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication (SIGCOMM)*, (Cambridge, Massachusetts, USA), pp. 95–109, Aug. 1999.

[89] A. Altin, B. Fortz, and H. Ümit, "Oblivious ospf routing with weight optimization under polyhedral demand uncertainty," *Networks*, vol. 60, no. 2, pp. 132–139, 2012.

[90] B. Fortz and H. Ümit, "Efficient techniques and tools for intra-domain traffic engineering," *International transactions in operational research*, vol. 18, no. 3, pp. 359–376, 2011.

[91] T. Erlebach and M. Ruegg, "Optimal bandwidth reservation in hose-model vpns with multi-path routing," in *23rd Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, (Hong Kong, China), Mar. 2004.

[92] L. Dai, Y. Xue, B. Chang, Y. Cao, and Y. Cui, "Integrating traffic estimation and routing optimization for multi-radio multi-channel wireless mesh networks," in *The 27th Conference on Computer Communications (INFOCOM)*, (Phoenix, AZ, USA), Apr. 2008.

[93] J. Wellons and Y. Xue, "Oblivious routing for wireless mesh networks," in *IEEE International Conference on Communications (ICC)*, (Beijing, China), May 2008.

[94] L. Dai, Y. Xue, B. Chang, Y. Cao, and Y. Cui, "Optimal routing for wireless

mesh networks with dynamic traffic demand," *Mobile Networks and Applications*, vol. 13, no. 1-2, pp. 97–116, 2008.

[95] J. Wellons and Y. Xue, "Towards robust and efficient routing in multi-radio, multi-channel wireless mesh networks," in *30th IEEE International Conference on Computer Communications (INFOCOM)*, (Shanghai, China), Apr. 2011.

[96] J. Wellons and Y. Xue, "The robust joint solution for channel assignment and routing for wireless mesh networks with time partitioning," *Ad Hoc Networks*, vol. 13, pp. 210–221, 2014.

[97] W. Wang, X. Liu, and D. Krishnaswam, "Robust routing and scheduling in wireless mesh networks under dynamic traffic conditions," *IEEE Transactions on Mobile Computing*, vol. 8, no. 12, pp. 1705–1717, 2009.

[98] C. Shepard, H. Yu, and L. Zhong, "ArgosV2: a flexible many-antenna research platform," in *Proceedings of the 19th annual international conference on Mobile computing and networking (MOBICOM)*, (Miami, USA), Sept. 2013.

[99] V. V. Vazirani, *Approximation Algorithms*. Springer Verlag, 2004.

[100] H. Wang, K.-W. Chin, R. Raad, and S. Soh, "A distributed maximal link scheduler for multi tx/rx wireless mesh networks," in *IEEE ICC*, (Sydney, Australia), June 2014.

[101] M. Kubale, *Graph colorings*, vol. 352. American Mathematical Soc., 1st ed., April 2004.

[102] E. R. Scheinerman, *Matgraph: a MATLAB toolbox for graph theory*. Department of applied mathematics and statistics, the Johns Hopkins University, Baltimore, Maryland, USA, 2008.

[103] 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, Mar. 2012.

[104] T. Salonidis and L. Tassiulas, "Distributed dynamic scheduling for end-to-end rate guarantees in wireless ad hoc networks," in *6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, (Urbana-Champaign, IL, USA), May 2005.

[105] J. R. Birge and F. Louveaux, *Introduction to Stochastic Programming*. New York, USA: Springer New York, second ed., 2011.

[106] D. M. Gay, *User's Manual for IBM ILOG CPLEX v12.1*. IBM, 2009.

[107] D. G. Luenberger and Y. Ye, *Linear and nonlinear programming*, vol. 116. Springer Science & Business Media, 2008.

[108] J. F. Traub and H. Woźniakowski, "Complexity of linear programming," *Operations Research Letters*, vol. 1, no. 2, pp. 59–62, 1982.

[109] D. Aguayo, J. Bicket, S. Biswas, G. Judd, and R. Morris, "Link-level measurements from an 802.11 b mesh network," *ACM SIGCOMM Computer Communication Review*, vol. 34, no. 4, pp. 121–132, 2004.