University of Wollongong

Research Online

University of Wollongong Thesis Collection 1954-2016

University of Wollongong Thesis Collections

2016

Novel centralized and distributed link schedulers for multi-transmit-receive wireless mesh networks

Yuanhuizi XU
University of Wollongong, yx879@uowmail.edu.au

Follow this and additional works at: https://ro.uow.edu.au/theses

University of Wollongong

Copyright Warning

You may print or download ONE copy of this document for the purpose of your own research or study. The University does not authorise you to copy, communicate or otherwise make available electronically to any other person any copyright material contained on this site.

You are reminded of the following: This work is copyright. Apart from any use permitted under the Copyright Act 1968, no part of this work may be reproduced by any process, nor may any other exclusive right be exercised, without the permission of the author. Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.

Unless otherwise indicated, the views expressed in this thesis are those of the author and do not necessarily represent the views of the University of Wollongong.

Recommended Citation

XU, Yuanhuizi, Novel centralized and distributed link schedulers for multi-transmit-receive wireless mesh networks, Doctor of Philosophy thesis, School of Electrical, Computer and Telecommunications Engineering, University of Wollongong, 2016. https://ro.uow.edu.au/theses/4759

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Novel Centralized and Distributed Link Schedulers for 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

Yuanhuizi Xu

Bachelor of Engineering (Telecommunications)

School of Electrical, Computer and Telecommunications Engineering

August 2016

Statement of Originality

I, Yuanhuizi Xu, 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

Yuanhuizi Xu

August, 2016

Abstract

Wireless Mesh Networks (WMNs) are an effective and attractive solution for broadband access in metropolitan as well as rural areas. They are used to deliver video, voice and data in indoor and outdoor environments. In WMNs, nodes are connected to one another wirelessly and they forward packets via multi-hop communications to each other or to gateways. WMNs technology has matured significantly over the past few years. A key advance in enabling higher capacity is to provide mesh routers with multiple transmit/receive capability to create so called Multi-Transmit-Receive (MTR) WMNs. This can be realized by equipping nodes with multiple directional antennas or using Multiple Input Multiple Output (MIMO) technologies. In particular, MTR WMNs enable nodes to transmit simultaneously to, or receive simultaneously from more than one neighboring node over the same frequency, and thereby yielding a WMN that has higher network capacity than conventional WMNs.

This capacity increase, however, is predicated on a link scheduler. Henceforth, the aim of this thesis is to design suitable link schedulers for this new form of WMNs. This thesis will focus on Spatial TDMA-based link schedulers. Their main goal is to derive the shortest possible superframe or link schedule that maximizes capacity or ensures all links are activated sufficiently often to meet their traffic demand. Critically, they will take advantage of the concurrent transmit or receive capability of routers. Also, they will consider the traffic load over each link. In particular,

links may require transmission times or have different queue lengths. This factor is significant as links with a high traffic load needs to be scheduled more frequently. Otherwise, data or packets will experience excessive delays or in the worst case, packet loss.

This thesis first contributes a scheduler that considers link weight or *air-time* corresponding to the time required to transmit a given traffic (or set of packets). The problem at hand is to activate the maximum number of active links at any point in time, and also to derive the shortest schedule that ensures all links are activated at least once for the duration of its assigned air-time. The proposed scheduler called A-TxRx first constructs the corresponding conflict graph of a MTR WMN and schedules links in a Maximal Independent Set (MIS). It then greedily adds links whenever a link finishes its transmission. As a result, unlike previous schedulers, links can start transmitting/receiving as soon as there is no conflict.

Most existing MTR link schedulers, including A-TxRx, are centralized. However, in practice a distributed version is more amenable to implementation. To this end, an important research question is how to design a distributed scheduler that fully utilizes the MTR capability of nodes. In order to solve this problem, this thesis proposes PCP-TDMA, a novel link scheduler that activates every link at least once within the shortest period of time in a distributed manner. Specifically, nodes change their allocated transmission slots gradually, and derive the MAX-CUT in each slot over time. Unlike centralized schedulers, PCP-TDMA does not require a central control node to gather global information nor require it to control the transmission schedule of all WMN nodes. On the contrary, nodes make their own decisions using *local* information.In addition, PCP-TDMA adapts to topological changes, and uses a simple method to adapt the current schedule quickly after a new node joins or an existing node dies.

The last contribution relates to queue stability. The problem is to design a distributed link scheduler that attains the maximal stability or capacity region of a network. In particular, any arrival rates within this region do not cause the

queues at nodes to grow to infinity. To date, no existing MTR schedulers ensure all queue lengths remain bounded at all times. To this end, this thesis contains a novel queue-aware distributed link scheduler. The key challenge is to stabilize all queues whenever the offered traffic load is within the stability region. This thesis first outlines a centralized throughput optimal scheduler, called LBC-ALGO. It is used to characterize the capacity region of different network topologies. However, LBC-ALGO requires instantaneous queue information to be sent to a central controller, which may be multiple hops away. Such requirement is impractical because doing so will incur excessive delays and signaling overheads. Henceforth, this thesis contains the first queue length aware link scheduler called dMaxQ. Each node requires only one-hop neighbors queue information and uses the celebrated max weight policy in a distributed manner. Specifically, dMaxQ attains the capacity or stability region by scheduling the highest weighted links in each transmission slot.

Acknowledgments

I would like to express my deepest gratitude to my supervisor, Associate Professor Kwan-Wu Chin, for being such a great mentor, philosopher and guide along the way. He has continuously helped me fill my knowledge gaps and sought opportunities to help me grow both professionally and personally. I truly appreciate the experience of being his student, and I believe that his extensive knowledge, extraordinary style, deep compassion and peculiar sense of humor have impacted me in a positive and great way.

I would also like to thank my co-supervisors, Dr. Raad Raad and Dr. Sieteng Soh, for their constant encouragement and valuable suggestions to help me develop and refine my research work.

Sincere thanks to the reviewers and editors of my conference and journal papers for their hard work, dedication, and constructive comments. Their feedback has definitely improved the quality of my publications.

Special thanks to my best friend Ethan, who supports me endlessly, cares for me deeply and makes me laugh pretty much every ten minutes. He is seriously the best companion I could have ever asked for.

Finally, heartfelt thanks to my family members. I want to thank my grandfather Lujia, who is the first engineer in my family. He sets a good example and taught me to live and work with integrity and the highest ethical standards. I also like

to thank my granny Shuqiao. She is always proud of me and prays that I can be happy and well every day. To my parents, Robert and Guangxia. There are no words to express my appreciation and gratitude for their unconditional love. They support me in anything that I want to do and that is how I achieve this milestone today. They have always been a warm comfort to me and did not pressure me in any way. Their enthusiasm for learning has driven them to continue to learn, grow and develop, which has been a huge source of encouragement and inspiration throughout my life.

Contents

List of Figures

List of Tables

Abbreviations

AP	Access Point
BER	Bit Error Rate
BS	Base Station
CBR	Constant-Bit-Rate
CSI	Channel State Information
DA	Directional Antenna
DEC	Directed Edge Coloring
DPC	Dirty Paper Coding
DSP	Digital Signal Processor
FEC	Forward Error Correction
GMS	Greedy Maximal Scheduling
GNSS	Global Navigation Satellite System
GPIM	Generalized Physical Interference Model
JRS	Joint Routing and Scheduling
LGS	Local Greedy Scheduler
LOS	Line-of-Sight
LP	Linear Program
MBAA	Multi-Beam Adaptive Array
MIMO	Multiple Input Multiple Output

MILP	Mixed Integer Linear Program
MINLP	Mixed Integer Non-Linear Program
MIS	Maximal Independent Set
MTR	Multi-Transmit-Receive
OPIM	Omni-directional Physical Interference Model
PAN	Personal Area Network
PNC	Piconet Controller
SINR	Signal-to-Interference-plus-Noise Ratio
SS	Subscriber Station
STDMA	Spatial Time Division Multiple Access
WLAN	Wireless Local Area Network
WMN	Wireless Mesh Network

Chapter 1

Introduction

1.1 Background

Wireless Mesh Networks (WMNs) are developing quickly and gaining a lot of attention because of their wide-ranging applications. They can be deployed in places such as homes, resorts, malls and hospitals [1]. In addition, they can be used to connect rural areas to cities to deliver voice and video [2]. A key advantage of WMNs is that wireless routers are self-configuring and self-organizing. Consequently, they can be quickly deployed to augment existing networks such as cellular and other forms of wireless networks such as IEEE 802.11, IEEE 802.15, and IEEE 802.16 networks [3]. For example, IEEE802.11s [4] is a standard that aims to use mesh technology to improve Wireless Local Area Networks (WLANs) [5]. This is particularly useful in homes [6] where a currently deployed access point (AP) is insufficient to cover all rooms. In another example, ZigBee or IEEE 802.15.5 WMNs [7] can be used to provide stable, scalable and interoperable Personal Area Networks (PAN) [5]. Similarly, IEEE 802.16e also supports mesh topologies, albeit optional. Apart from that, vendors such as Firetide [8] and Aruba [1] sell WMNs products that extend the coverage of WiFi access points, surveillance cameras and other devices to better serve heterogeneous user demands [9].

The capacity of WMNs is a significant issue, especially if a communication backbone is constructed using a WMN. Unfortunately, Gupta et al. [10] showed that in a wireless network with n nodes, the throughput of each node is $\Theta(\frac{W}{\sqrt{n}})$ bit-meters per second of data, where W is the data rate. This means as we scale the network, i.e., increase n, throughput tends to reach zero. This is because the channel is shared with increasing number of nodes.

To improve capacity, a promising approach is to equip nodes with Multi-Transmit-Receive (MTR) capability. Specifically, nodes are equipped with multiple Directional Antennas (DAs) or adaptive arrays. This means all nodes can transmit or receive simultaneously from their respective neighbors; see Figure 1.1(a)(b). Consequently, unlike conventional wireless routers that can only carry out *one* omni or directional transmission/reception, see [11][12][13][14][15], the nodes in MTR WMNs have a higher network capacity because they can transmit/receive k distinct packets to/from their neighbors; in Figure 1.1, we see that node A transmitting simultaneously to neighbors C, D and B. This is then followed by node A receiving a packet from each of the said neighbors simultaneously.

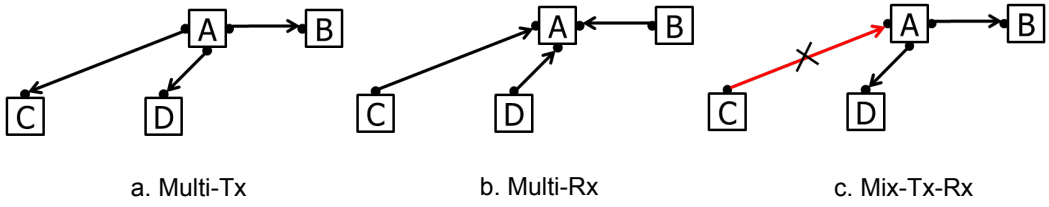


Figure 1.1: Different phases of MTR transmissions/receptions: (a) transmission, (b) reception, (c) invalid reception; i.e., no transmit *and* receive in the same phase.

In MTR WMNs, the state of each node is separated into two phases: transmission and reception. At any point in time, the node is either in the transmission phase, see Figure 1.1(a), or the reception phase, see Figure 1.1(b), while its neighbors are in the opposite phase. However, as shown in Figure 1.1(c), a node transmitting and receiving concurrently, which is called Mix-Tx-Rx state, is not feasible. This is because the reception of node A is affected by its own transmissions due to self-interference or side-lobes [16]. It is worth emphasizing that all transmissions/receptions are con-

ducted on the same frequency. Hence, a *key constraint* when scheduling links in MTR WMNs is that nodes must not transmit *and* receive simultaneously[1].

To date, there are three main realizations of MTR WMNs. First, in [16], the authors demonstrated an MTR WMN using off-the-shelf routers equipped with multiple radios and low cost 24 dBi parabolic grid antennas that operate on a *single* frequency. Nodes disable their carrier sense to allow concurrent transmissions, and transmit power control is used to ensure incoming links have sufficient signal strength to ensure correct reception. Another example is to equip nodes with 60 GHz radios and antenna arrays. With its high directivity, the resulting 60 GHz wireless links can be regarded as pseudo-wires, and interference from neighboring transmissions can be ignored [18]. Lastly, MTR can be achieved via Multiple Input Multiple Output (MIMO) technologies [19]. Routers are assumed to have channel state information (CSI). This assumption is reasonable given that nodes are primarily static and pilot symbols can be transmitted periodically to learn the CSI. The resulting system is also called multi-user MIMO (MU-MIMO) in the literature [21]. Note that it is now practical to equip routers with a high number of antenna elements or radios. A practical example is Argos [20], where the authors demonstrated a system with 64 antenna elements per node. Consequently, Argos allows the creation of a practical MTR WMN where a node can transmit to or receive from 64 neighbors simultaneously. With sufficient antenna elements or degree-of-freedom, each node is thus able to null out interfering transmissions or null interference to nodes that are receiving data.

1.2 Motivation and Problems

The maximum capacity of a MTR WMN is tied closely with the deployed link scheduler. In this regard, the fundamental problem is to determine which links

[1] As an aside, in [17] and references therein, researchers have shown the possibility of removing self-interference to allow concurrent transmit *and* receive over the same frequency. Hence, as a future work, it will be interesting to address the problems in this thesis in full-duplex MTR WMNs.

transmit at what times. The key challenge is ensuring these links are interference-free and activated sufficiently often to satisfy traffic demands. To this end, this thesis proposes and studies link schedulers that use Spatial Time Division Multiple Access (STDMA) [22]. As an example, consider Figure 1.2. We see a link schedule or *superframe* for a MTR WMN that allows all links a transmission opportunity. The superframe consists of two slots and thus has length two. Observe that each slot contains non-interfering links and ensures a node does not transmit *and* receive simultaneously; these links are indicated as solid and dash arrows respectively. After the superframe is generated by the scheduler, it is repeated periodically. To ensure the highest possible capacity, it is critical that this period is short; conversely, the rate in which links are activated is high. Thus, the aim of such schedulers is to minimize the superframe length. Moreover, each slot must contain the maximum possible number of non-interfering links. Apart from that, ensuring queues remain stable is also of concern. This minimizes delays and ensures packets are not loss due to congestion.

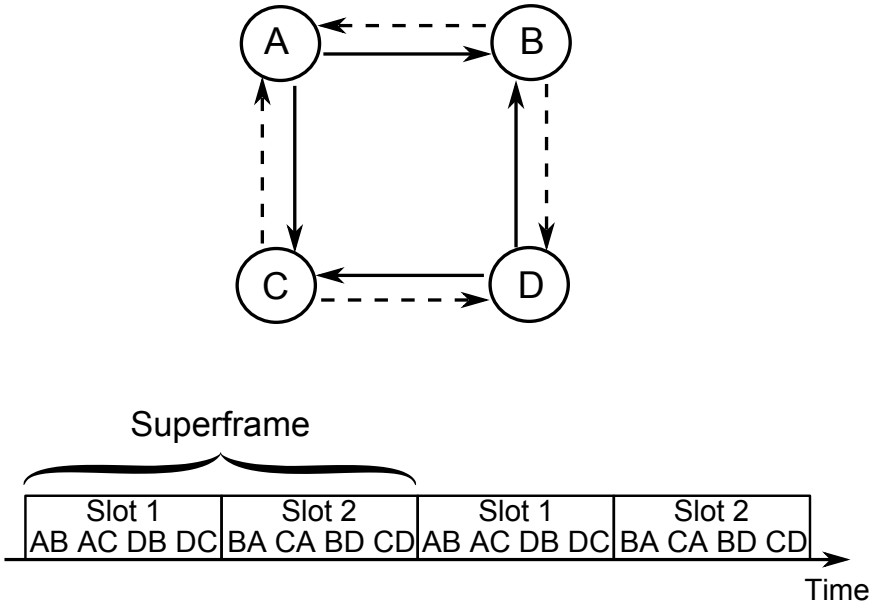


Figure 1.2: An example MTR wireless network and its TDMA schedule

1.2.1 Link Scheduling with Air-Time

When scheduling links, it is of great importance to consider links with different weights; e.g., links with varying activation duration. This is necessary because of two reasons. Firstly, in practice, it is likely that links will have different loads. For example, links preferred by many nodes or a router may be serving an area with a large number of subscribers. Secondly, different data rates may be used by links in order to counter the vagaries of the wireless channel. For example, IEEE 802.11a supports data rates up to 54 Mbps. This means, for a given packet size, the air-time required to transmit said packet will vary depending on the data rate or channel condition as well as the number of packets to be transmitted.

To date, see Chapter 2, although some existing MTR schedulers have considered traffic demands, they do not consider links starting at different time points. Consider Figure 1.3. We see an MTR WMN with three nodes connected in a clique. The number next to each link represents its required transmission or air-time. Also shown in Figure 1.3 is the schedule for each link derived using 2 Phase (2P) [16]. In this example, at time $t = 0$, link AB and AC start to transmit simultaneously in the first slot for 10 units of time. Then at $t = 10$, the corresponding links BA and CA are activated in the second slot for the duration of five units only after both AB and AC finish. At $t = 15$, the activation of link BC and link CB takes place in the third and fourth slot for a duration of nine and three time units respectively. Thus, for this example, 2P derives a superframe that has length $T = 27$. In this example, we see that links remain active in SynTx or SynRx for the same duration regardless of the actual link weight. This situation leads to the following problem. Consider link AB and AC, which are activated simultaneously in the first slot. The slot length needs to be the longer air-time, which is 10. Notice that link AB finishes in one time unit, causing 9/10 of the link capacity to be wasted, leading to a lower throughput. This is because no other links are scheduled until link AC finishes. In fact, if node B wants to transmit to C, it should be able to initiate transmission at

any time after the first 1/10 of the slot. If link BC activates at an earlier point of time, throughput improves.

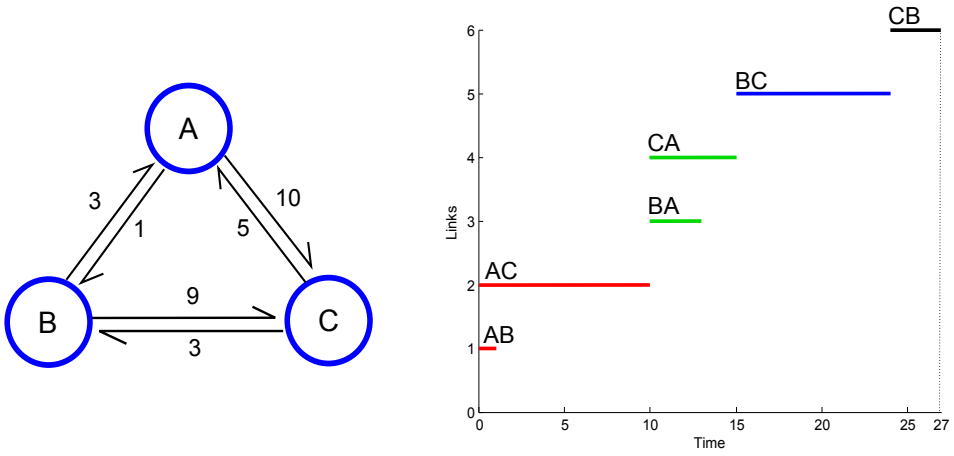


Figure 1.3: (a) An example network where links have different air-times; (b) Link scheduling using 2P.

1.2.2 Distributed Link Scheduling

The problem of calculating the shortest superframe whilst maximizing the number of concurrent links in each slot is equivalent to the NP-complete, MAX-CUT problem [23][24]. Most of the link schedulers proposed in the literature are centralized and are run by a controller; e.g., one located at a gateway. This means a central controller must collect required information from every node, compute a feasible schedule for the entire network, and disseminate the computed schedule to the relevant nodes before it can be used. Thus, they are only practical for small WMNs. For large-scale WMNs, there exist two critical limitations of central controllers/schedulers. Firstly, the computation complexity increases exponentially with network size. Secondly, centralized solutions lead to excessive signaling overheads, large propagation and contention delays. That is because a large portion of the network capacity is used to gather information required to construct a schedule and disseminating the computed schedule. Moreover, central controllers/schedulers are ill-equipped to adapt

to topology changes and is a point of failure.

The aforementioned limitations thus motivate the development of distributed link schedulers. In other words, nodes only communicate with their one-hop neighbors and compute the local link schedule for their incident links. To this end, a key research question is how to design a distributed scheduler where all nodes in the network work collaboratively with their immediate neighbors to generate an optimal superframe.

1.2.3 Distributed Queue Aware Link Scheduling

The queue length of each link is an important parameter and needs to be taken into consideration when scheduling links. To date, as will be discussed in Chapter 2, most MTR schedulers aim to minimize the superframe length. A key observation is that none of them have considered queue length, which may lead to two critical consequences. The first one is *slot under utilization*, meaning a scheduled link has no packet to transmit. The second is *network instability*, where queues grow continuously causing packet losses and large delays. The challenge is thus to design a throughput optimal scheduling policy.

In [25], the Max Weight, or Max Scalar policy is proved to be throughput optimal because it can achieve the largest possible *capacity region* (also called *stability region*), which is defined as the set of arrival rate vectors under which queues are stable; i.e., they do not grow to infinity. However, this policy is centralized and requires instantaneous queue length information, which is not practical in WMNs. In particular, the centralized policy will incur many rounds of requests, and signaling overheads as well as propagation delays, which result in stale queue size information. Consequently, a distributed algorithm that uses only local information, i.e., one or two hop neighbors, is highly desirable.

To date, a number of studies have focused on designing low-complexity and low-overhead distributed scheduling policies. For WMNs where the queue length of each

link is known as a link weight, designing a distributed throughput optimal algorithm that stabilizes the network under the *maximum capacity region* is critical to applications such as video and voice. However, the capacity region of the existing queue aware distributed scheduling policies is typically smaller than that of centralized solutions. This limitation thus motivates the design of a distributed algorithm that is throughput optimal. Henceforth, a key problem in this thesis is to seek a scheduler that approximates the throughput achieved by the centralized policy. Notably, no such schedulers exist for MTR WMNs.

1.3 Contributions

In this thesis, novel scheduling algorithms are proposed to solve the foregone problems. In a nutshell, it contains the following contributions.

1.3.1 Scheduling links with varying air-time

This thesis outlines A-TxRx, the first centralized MTR link scheduler that considers links that require different transmission times. In particular, it considers the assigned air-time of links in order to meet the underlying link demand. A-TxRx derives the shortest feasible superframe subject to the Mix-Tx-Rx constraint. According to extensive simulation, the results show that A-TxRx has better performance in all considered scenarios; i.e., an average of 40% shorter superframe length than 2P, especially when the network is fully connected. The superframe length of A-TxRx is at most 70% shorter than state-of-the-art approaches. Additionally, an improvement to A-TxRx is outlined, where scheduled links are added opportunistically to further increase network capacity. The results show that the number of concurrent transmitting links when running A-TxRx to be 40% more than JazzyMAC [26] on average. Furthermore, this thesis analyzes and proves that A-TxRx is a collision free scheduler for arbitrary topologies and has a running time of $\mathcal{O}(|V|^5)$ for WMN with $|V|$ nodes.

1.3.2 Distributed TDMA Link Scheduler

This thesis presents a distributed link scheduler called Period Controlled Pseudo-TDMA (PCP-TDMA) which considers the MTR-capability of nodes. Specifically, nodes gradually derive the links to be activated in each slot over time. It is worth noting that most existing distributed links schedulers are designed under the assumption of the k-hop, protocol or physical interference model. To illustrate how PCP-TDMA derives a superframe or schedule, consider Figure 1.4. We see that the initial superframe $\mathcal{SF}_a$ has a length of four slots. All links are activated as per the no mix-Tx-Rx constraint. Over time, the links, e.g., e_{BC} and e_{CB}, change their transmission slot with the goal of reducing the superframe length; i.e., allocating e_{BC} into slot s_2 of $\mathcal{SF}_{a+x}$ and e_{CB} in slot s_1 of $\mathcal{SF}_{a+x+y}$ shortens the superframe length to two.

In a nutshell, PCP-TDMA addresses the following features. Firstly, nodes improve their reserved random slots over time without causing collisions. Secondly, nodes determine the last reserved slot and then update their period to the same value. In addition, several properties of PCP-TDMA are analyzed, including the configuration of the initial period, and the correctness and convergence of the algorithm. Compared to the state-of-the-art, namely a centralized algorithm called ALGO-2 [23], and two distributed approaches, namely JazzyMAC [26] and ROMA [27], the results show that PCP-TDMA achieves similar performance to ALGO-2, and outperforms JazzyMAC and ROMA significantly. Specifically, in a fully connected network, the superframe produced by PCP-TDMA is respectively only $\frac{1}{3}$ and $\frac{1}{2}$ the length of that generated by JazzyMAC and ROMA.

1.3.3 Distributed Max Weight Link Scheduler

As mentioned, a queue-aware scheduler is required to avoid large delays and packet loss. Critically, there is a lack of distributed MTR scheduler that considers queue length and is throughput optimal. Henceforth, this thesis proposes dMaxQ, the first

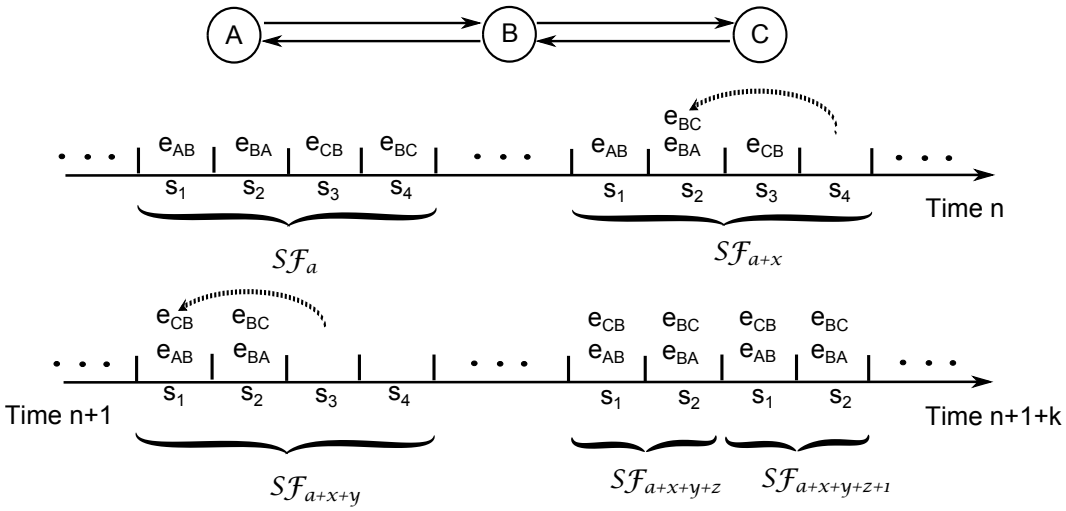


Figure 1.4: Example topology and schedule. The final superframe has a length of two slots

distributed, throughput optimal link scheduler for MTR WMNs. In fact, dMaxQ is the first distributed MAX-CUT algorithm that incorporates the max weight policy to achieve throughput optimality. Specifically, dMaxQ provides a deterministic method that uses only local queue length information to schedule the highest weighted links in each time slot. For comparison purposes, a centralized, greedy scheduler called LBC-ALGO is outlined to characterize the capacity region of different network topologies. The performance of dMaxQ is evaluated in both single-hop and multi-hop traffic models, and is compared against other approaches including two queue length aware centralized algorithms, and state-of-the-art distributed approaches: JazzyMAC and ROMA. The results show that for single-hop and multi-hop traffic scenarios, dMaxQ obtains, respectively, 100% and 90% of the throughput received by the theoretical, centralized policy. It is worth mentioning that dMaxQ achieves the same capacity region as the centralized algorithm in all single-hop scenarios, indicating it is a throughput optimal solution. Other distributed algorithms such as JazzyMAC only managed 25% of the theoretical throughput.

1.4 Publications

The aforementioned contributions have resulted in the following articles:

1. Y. Xu, K-W Chin, S. Soh and R. Raad. **Scheduling Links with Air-Time in Multi Tx/Rx Wireless Mesh Networks**, Springer Wireless Networks (WINET), 22(6), pp.1999-2012, June, 2016.

2. Y. Xu, K-W Chin, S. Soh and R. Raad. **A Novel Queue Length Aware Distributed Link Scheduler for Multi-Transmit/Receive Wireless Mesh Networks**, IEEE 15th Intl. Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Sydney, Australia, June, 2014.

3. Y. Xu, K-W Chin, S. Soh and R. Raad. **A Novel Distributed Max Weight Link Scheduler for Multi-Transmit/Receive Wireless Mesh Networks**, IEEE Transactions on Vehicular Technology (TVT), 2016. Accepted. *Available online: http://dx.doi.org/10.1109/TVT.2016.2516979*

4. Y. Xu, K-W Chin and S. Soh. **A Distributed Pseudo TDMA Protocol for Multi-Transmit-Receive Wireless Mesh Networks**, IEEE 19th International Symposium on Wireless Personal Multimedia Communications (WPMC), Shenzhen, China, November, 2016. *Under Review*

1.5 Thesis Structure

1. *Chapter 2.* This chapter surveys works related to centralized and distributed link scheduling approaches for MTR WMNs.

2. *Chapter 3.* This chapter presents A-TxRx, a centralized algorithm that considers varying transmission times. It also shows the superframe bound for various topologies.

3. *Chapter 4.* This chapter proposes a simple, iterative, distributed Medium Access Control (MAC) called PCP-TDMA to construct superframes with minimal length.

4. *Chapter 5.* This chapter introduces dMaxQ, a novel queue length aware distributed link scheduler that employs the max weight policy.

5. *Chapter 6.* This chapter concludes the thesis, provides the main contributions and directions for future research.

Chapter 2

Literature Review

This chapter reviews prior works on TDMA-based link schedulers that take advantage of nodes equipped with directional antennas and those with MTR capability. In addition, it will survey queue-aware scheduling algorithms. A summary is then provided in Section 2.4.

2.1 Nodes with Directional Antennas

Nodes with a Directional Antenna (DA) or smart antenna are able to direct their transmissions toward one or more geographical areas [28]. Traditional directed antennas use mechanical rotation, and have been used in works such as [29], [30], [31] and [32]. On the other hand, smart antennas make use of a number of radiating elements and use a Digital Signal Processor (DSP) as a control unit. There are three types of smart antennas [33]: 1) switched beam with predetermined antenna radiation patterns; example of which include [34], [35] and [36], 2) steerable, single beam with nulling capability, where the main beam is directed towards a user and nulls are formed to remove interfering transmissions; example works include [37] and [38], and 3) adaptive array antennas with beamforming and signal suppression capabilities. They are used in works such as [39], [40] and [41].

The use of DAs is crucial in WMNs. Zhang et al. [42] compare the capacity

of WMNs using omni and directional antennas. Through exhaustive analysis, the authors made the following observations. First, the capacity depends on the ratio of interference to the transmission distance. Secondly, WMNs have higher capacity when using DAs. In addition, the capacity increases with the number of antennas m and smaller beamwidth θ. Note that as θ decreases, the capacity reaches an upper bound of $\frac{Cm}{N}$, where C is the channel bandwidth, N is the number of non-gateway nodes. Yi et al. also pointed out in [43] that the use of DAs yields $\frac{4\pi^2}{\alpha\beta}$ more capacity in random ad hoc networks as compared to using omni-antennas, where α and β are the beamwidth of the transmitter and the receiver, respectively.

The higher capacity resulting from the use of DAs are due to the following reasons [44]. Firstly, DAs allow nodes to focus their transmission power onto a desired direction as opposed to omni-antenna transmission where the power is equally radiated in all directions. Moreover, DAs provide greater transmission range, and thus fewer hops are required between source and destination nodes. In addition, the directionality afforded by DAs improves spatial reuse and helps mitigate interference.

A number of works have sought to exploit the aforementioned advantages of DAs. They are categorized as random access and TDMA based scheduling protocols. Random access MAC protocols include examples such as [32] and [45]; and tone-based protocols [46] [47]. These random access protocols address the hidden terminal and deafness problem. However, random access MACs are not collision-free. They do not achieve high capacity, and they lack delay bounds. On the other hand, TDMA scheduling approaches pre-arrange and guarantee collision-free transmissions. Moreover, TDMA MAC protocols can better exploit spatial reuse to allow concurrent communications between nodes and thus deliver higher network throughput. To this end, this thesis will only focus on papers that propose TDMA-based MAC protocols.

In [48], Masri et al. consider routers equipped with M directional antennas. Transmissions and receptions can occur directionally or omni-directionally. Time is divided into slots that consist of a control and transmission part. The control part is

further divided into N mini-slots, where N is the number of nodes within two hops. This protocol obtains full knowledge of concurrent transmissions in the vicinity of each mesh router. This information is then used to design a fair and collision-free time slot assignment algorithm for each router. Nodes are first assigned to mini-slots subject to the constraint that any two nodes within two-hops are assigned a separate mini-slot. In the second step, each mini-slot is further divided into three parts. Assume a node S has a packet to transfer to a node R with a known location, and S is assigned to the first mini-slot. It broadcasts a jamming signal omni-directionally, which is a busy-tone signal that contains no information. Neighbors that detect the jamming signal note that node S will be busy in the next transmission part. In the second mini-part, node S transmits a jamming signal directionally to node D; this signal informs D it is the destination node. During the third mini-part, node D broadcasts a jamming signal omni-directionally to inform other nodes that it will be busy receiving.

Paso et al. [39] schedule links using a TDMA based Slot Borrowing (TDMA-SB) scheme. This scheme maximizes network capacity because it utilizes any unused time slots, i.e., they are assigned to nodes with high traffic demands or those with strict delay requirements. The authors assume that each node is equipped with a smart antenna that sends control messages omni-directionally and sends data packets directionally. In addition, nodes are equipped with Global Navigation Satellite System (GNSS) receivers and electronic compasses, meaning they have location information and also antenna direction. When an unused slot is detected, TDMA-SB works as follows. Initially, all nodes send their queue length to their neighbors to determine transmission priority. Here, the highest priority goes to node S_1 that has the longest queue. Then S_1 notifies its neighbors that the first packet in its queue will be transmitted to the destination node D using this slot. After that, node S_2 with the second longest queue performs an interference check to see if it is able to perform transmission in the same slot without interfering with the transmission of S_1. If so, S_2 notifies its neighbors. The remaining nodes then perform an interference

check according to their weight or priority.

In [34], the authors propose a distributed TDMA-based MAC protocol to increase network throughput as well as reduce average delay. Hao et al. assume that every node is equipped with a switched-beam antenna that provides two operation modes: omni and directional. All control messages are transmitted omni-directionally, and data is transmitted directionally. Every node maintains an information table that records the unique MAC ID and GPS coordinates of all nodes. Each superframe consists of n time slots, where n is the number of nodes. In the i-th slot, a node with ID i is regarded as the *Main* node, while other nodes are called *Normal* nodes. Every time slot is partitioned into three stages. The first stage is called "Main Node Statement", where the *Main* node broadcasts a RTS message containing the ID and coordinates of the source and destination of the packet to be sent. The next stage is the "Contention Stage", in which Normal nodes use the binary countdown mechanism of [49] to prevent collisions. Each Normal node creates a unique binary sequence based on its ID, packet urgency and waiting time. The sequence of contending nodes is sent bit by bit. Each bit is OR-ed together with that of other nodes by the channel. A Normal node quits channel contention as soon as it notices that its zero bit becomes a one. At the end of the Contention stage, the winning node sends its RTS message to all neighbors. In the final "Data Transmission Stage", data packets are transmitted directionally.

In [37], Ramamurthi et al. study link scheduling and power control. All nodes are equipped with a switched beam antenna that has a beamwidth of 30°. To maximize throughput, the authors define the problem as a Mixed Integer Linear Program (MILP) that incorporates the Generalized Physical Interference Model (GPIM). It is more realistic than the protocol interference model used in [10] because it extends Omni-directional Physical Interference Model (OPIM) with directional transmissions. For example, a transmission on link (v_s, v_d) is interfered by another simultaneous transmission on (v_i, v_j) if the distance between v_d and v_i is smaller than a certain threshold. The MILP models the location of mesh routers, gateways, traffic

demand and paths. Its outputs are the link schedule and corresponding transmit power of each link. To reduce computational complexity, the authors provide a heuristic. It first sorts all links based on interference, which is shown as edges in a conflict-graph. The second step is to schedule the sorted links. That is, at the beginning, the link with the most edges in the conflict-graph is scheduled in as many time slots as it requires. Then the rest of the links are scheduled based on two admissibility criteria: (i) primary interference does not appear in any time slot when scheduling a link, and (ii) the Signal-to-Interference-plus-Noise Ratio (SINR) experienced by any node is within a given threshold.

In [50], the authors propose a QoS-aware link scheduler for real-time constant-bit-rate (CBR) traffic. Similar to [51], they consider a tree topology where multiple Subscriber Stations (SSs) connect to one Base Station (BS) through one or more hops. A single radio is equipped in each station. However, instead of assuming multiple channels as in [51], the authors assume only a single channel. They also assume link rates, routing table, interference model, and network topology are known a priori. They then formulate the scheduling problem as an ILP. However, the resulting ILP is computationally intractable. Consequently, they propose a heuristic scheme called Bottleneck First Scheduling (BFS). The main idea is: (i) scheduling nodes with higher traffic demand earlier than those with a lower traffic demand, (ii) at each node, schedule packets that need to be forwarded through more hops to their destination earlier than those needing fewer hops.

Reference [30] is the first work that addresses the problem of generating a conflict-free schedule with the maximum end-to-end delay as a constraint. In other words, the schedule guarantees that no flows violate their delay bound. The authors assume that the delay of a flow is a function of: (i) queuing delay, which is determined by the number of flows using the same link, and (ii) scheduling delay, which is decided by the transmission order of incoming and outgoing links. The authors formed an optimization problem with the objective of finding the minimum superframe length, link activation time and transmission duration subject to delay constraint.

However, this problem is non-linear and non-differentiable due to its delay bound expression, which makes this problem very hard to solve. Thus, the authors propose a heuristic solution that considers queuing and scheduling delay separately. This solution iteratively carries out the following two stages: (i) solve an ILP to obtain the minimum scheduling length and scheduling delay, and (ii) choose the flows with a queuing delay higher than a given threshold. Then in the next iteration, these chosen flows are assigned a higher weight in the ILP in order to obtain more time slots. The iteration continues until the delay constraint of all flows is met.

There are several cross layer approaches that consider joint routing and scheduling (JRS) for multi-hop WMNs. In [31][40][29] and [52], the routing paths are not given in advance, which will have an impact on the final link schedule. Thus, the authors propose joint approaches where the scheduler considers the routing path between a given source and destination pair whilst deriving a link schedule. In [31] Capone et al. address the joint routing and scheduling problem in WMNs where nodes are equipped with DAs. In addition, they also consider power control and rate adaptation. The authors assume that only a single channel is available. Each node is equipped with one DA that has a main lobe with an angular width of 45°. The main and side lobes have a gain of 10 dBi and 0.8 dBi, respectively. Capone et al. adopted the SINR interference model; i.e., it takes into account the cumulative effect of several interfering transmissions. The main problem addressed is to determine the optimal routing paths that minimize the superframe length. To solve this problem, the authors presented a column generation-based heuristic that adheres to the following constraints: (1) flow conservation, where the incoming and outgoing traffic flow must be equal, (2) each link must be activated in at least one slot for each packet to be sent, (3) half-duplex constraints, whereby incoming and outgoing links of the same node cannot be activated simultaneously, (4) transmission quality, as determined by SINR and transmission power. The researchers defined a comparison metric called ψ to be the ratio between the superframe length and the total number of packets to be routed. Their results show that the ψ of a system using DAs is 45%

less than one that uses omni-directional antennas.

Spyropoulos et al. [40] proposed an energy-efficient routing and scheduling algorithm for large scale ad-hoc networks that consist of hundreds or thousands of nodes. On each wireless node, there is a single electronically steerable DA with an antenna gain of 2 or 3 dBi. For simplicity, the DA radiation pattern is assumed to have only one main lobe, while side lobes are ignored. This algorithm has four steps. The first step is least cost routing. Initially, the Dijkstra algorithm is used to derive the least cost path for each node. The authors consider two metrics in the routing step. One is to minimize the energy consumed per packet. The other is used to maximize network lifetime. Specifically, the protocol will avoid routing traffic through nodes with low energy. The second step is link flow matrix calculation. This is done by establishing a matrix $F' = \{f'_{ij}\}$ in which each element indicates the traffic between node i and j. In the third step, a variable called uptime, denoted as w_{ij}, is used to represent the activation duration of a link between node i and j. The uptime is calculated for each link as follows: given a node i with three neighbors, namely j, a, and b, then $w_{ij} = \frac{f'_{ij}}{f'_{ij}+f'_{ia}+f'_{ib}}$. After the uptime is assigned to each corresponding link, the final step is to schedule links based on their uptime in order to get the shortest superframe. The links with equal or similar uptime are grouped together. The results obtained showed that the lifetime of networks using DAs is two to four times more than those that use omnidirectional antennas. In addition, using energy-aware routing improves lifetime by 45% as compared to conventional routing schemes such as minimum hop routing.

Cappanera et al. [29] proposed a joint routing and link scheduling approach for WMNs carrying real-time traffic. The aim is to minimize delay so that real-time traffic meet their deadline. The authors formulated a mixed integer-nonlinear program (MINLP), in which the decision variables include the activation time of links, transmission order and transmission duration. The MINLP considers interference, flow conservation and actual traffic load constraints. However, the resulting MINLP can only be used for small networks. For larger networks, the authors remove the

transmission order from the decision variables of the MINLP. Instead, they first determine the transmission order using a conflict graph and a general K-coloring method [53]. Then they apply Lagrangian relaxation to decouple the link scheduling and routing constraints from the MINLP. Thus, the MINLP is separated into two smaller problems; both is then solved using a standard LP solver.

In [52], the authors formulate a joint routing, channel assignment and link scheduling (RCL) problem for multi-channel, multi-radio WMNs to maximize overall network throughput. Specifically, if $l(u)$ is the aggregated traffic load from all nodes that are associated with node u, then the goal is to allocate the minimum bandwidth $\lambda l(u)$ to satisfy given traffic demands. In addition, for each node u, traffic demands are routed in proportion to $l(u)$ in order to prevent starvation of nodes that are far from gateways. They propose a centralized approximation algorithm to maximize λ subject to fairness constraint. The algorithm consists of four stages. In the first stage, the algorithm assigns the i-th available channel to the i-th radio interface of each node. In the second stage, flows are diverted through other links to ensure a feasible channel assignment. Next, each flow is re-distributed in the flow graph in order to achieve minimum interference whilst keeping the network throughput the same. For example, if a flow is experiencing interference, the flow value can be distributed to other channels. The last stage is allocating links into a schedule S that consists of a maximum of $K \times c(q)/I$ time slots, where K is the number channels, $c(q)$ is an interference constant, and I is the number of radio interfaces.

Prior studies have focused on designing DAs based MACs for WMNs. Their goal is to schedule all directional links efficiently, i.e., without interference or collision, within the minimum time duration, which in turn maximizes network throughput. The main method is to assign interfering links into different time-slots to achieve time domain separation. However, no studies have considered the possibility of multiple concurrent transmissions or receptions by a node. Specifically, MTR WMNs employ multiple DAs or adaptive arrays to increase network capacity. In particular, all nodes can transmit simultaneously to, or receive simultaneously from a number of

neighboring nodes, and thereby yielding a WMN that has higher network capacity than those that use an omni-directional or a single directional antenna.

2.2 Multi Transmit/Receive WMNs

2.2.1 Prototypes

Before delving into MTR link schedulers, this section first surveys MTR implementations/prototypes. References [16][14] and [54] introduce MTR prototypes that use distinct radios with a directional antenna. In [16], the authors use a MTR WMN to interconnect rural villages in India. Each off-the-shelf router (or node) has multiple radios. Each one is connected to a low cost 24 dBi parabolic grid antenna. This allows routers to establish long distance IEEE 802.11 links in a 44 km^2 area. Each antenna has a beamwidth of 30°. To achieve simultaneous transmission or reception at each router, Raman et al. [16] made several innovations. First they modify CSMA/CA, which includes: (i) disabling immediate ACK. This ensures an ACK does not collide with a node's reception on an adjacent radio, (ii) modifying the carrier sense mechanism to enable simultaneous transmissions. This is required because the side-lobes from each antenna causes an adjacent radio to detect a carrier and hence, stops it from transmitting. The second innovation is spacing the angle between two neighboring links to be greater than 30° because smaller angles cause higher side-lobe levels. This ensures that the Bit Error Rate (BER) of each link is negligible ($< 10^{-6}$), as well as the SINR to be at a high enough level: 14 to 16 dB. The third innovation is to place radios that are on the same node four to five metres apart from each other to avoid near-field effect.

Patra et al. [14] presented an MTR WMN called WiFi-based Long Distance (WiLD). Nodes are equipped with Atheros IEEE 802.11a/b/g radios. There are seven nodes that form six long distance links (ranging from 1 to 45 km). Each node is a 266 MHz X86 Geode single board computer running Linux 2.4.26. These

low cost computers are equipped with off-the-shelf high-power IEEE 802.11 wireless cards, and commercially available 24 dBi directional antennas with beamwidth 8°. Moreover, there are one to five radios on each node. Patra et al. modified the IEEE 802.11 MAC as follows: (i) link-layer retransmissions, where ACKs and CSMA are disabled to allow synchronous transmission/reception; similar to the reasons outlined in [16], (ii) the value of DIFS is increased to allow for longer propagation delays, and (iii) concurrent transmission and reception are not allowed because of sufficiently high gain, 4-8 dBi, side lobes. Thus, in [14], a basic TDMA mechanism that uses loose synchronization is employed instead of CSMA/CA to avoid inter-link interference, and a combination of Forward Error Correction (FEC) and bulk ACKs to reduce signaling overheads and improve link utilization.

Dutta et al. [54] outlined a MTR WMN with full duplex links. To establish long-distance links ($>$ 25 km), the researchers employed 24 dBi Andrew made antennas with 8° beamwidth. These high gain parabolic grid directional antennas are placed on top of five to seven story buildings with an angular separation of more than 30° between neighboring links. The links are established using Prism2 chipset based Senao NL-2511CD plus ext2 PCMCIA cards that support the IEEE 802.11b standard. These low-cost off-the-shelf WiFi cards are housed in Linux based PCs and laptops. Moreover, they used the open source HostAP wireless driver [55]. Dutta et al. model the network as a graph $G = (V, E)$, where V is the set of nodes and E is the set of directed links. Each link is established between a pair of nodes by aligning their respective directional antennas. For a given node, they assign two non-overlapping channels to the incoming and outgoing links, respectively. Since these two channels are non-overlapping (orthogonal) and thus non-interfering, it is possible for all the edges (links) at a node to operate in full-duplex mode at all times.

MTR can also be achieved using advances in multi-user MIMO (MU-MIMO) [56] [57] [58]. MU-MIMO has the following advantages. First, it improves channel gain via spatial multiplexing and thereby allowing the antenna arrays to use higher

data rates, and communicate with spatially disparate users simultaneously. Second, it improves robustness via spatial diversity. This is because MU-MIMO is more resilient to propagation characteristics such as channel rank loss and antenna correlation. Third, by using MU-MIMO, the spatial multiplexing gain of the base station can be directly obtained without the need for multiple antenna terminals. Thereby, the development of small and low-cost terminals is made possible by keeping the intelligence and cost on the infrastructure side.

Realizing a MU-MIMO network, however, requires the following challenges to be addressed. Specifically, transmitters require perfect Channel State Information (CSI). For example, in [58], Bahadori et al. proposed a MU-MIMO WMN comprising of nodes with a linear antenna array. The antenna elements are half-wavelength dipoles, and are placed uniformly by a quarter of wavelength from each other. Every node has the same number of antennas, which ranges from three to nine in different experiments. The resulting system provides a significant boost in capacity as expected. The interference is effectively reduced by beamforming and interference cancellation techniques applied by receivers.

Another practical MU-MIMO design is called per user unitary and rate control (PU2RC) [59]. It supports multiuser simultaneous transmissions, enables quantized CSI feedback, and exploits multiuser diversity. In PU2RC, the fundamental constraint is the orthogonal beamforming feature, whereby each user selects a beamformer from a codebook of multiple orthonormal bases. The major strength of this method is that compared with the optimal multiuser transmission strategy called Dirty Paper Coding (DPC) [60], and a close-to-optimal solution Tomlinkson-Harashima [61], PU2RC is less sensitive to CSI inaccuracy and is thus practical.

Lastly, MTR also can be realised using 60 GHz radios [18] [62] [63] and an antenna array. The unlicensed 60 GHz "millimetre (mm) wave" band has many advantages. Namely, it has a wide, continuous spectrum ranging from 57-64 GHz, which supports multi-gigabit wireless transmissions; its oxygen absorption characteristics mean high attenuation and consequently helps promote high spatial reuse,

and more importantly, compared with lower frequency bands, transmissions have high directivity gain thanks to the shorter wavelength. All the above features allow nodes to achieve gains of around 25 dBi easily, and thereby wireless links can be regarded as pseudo-wires because they do not cause interference with one another.

2.2.2 Link Schedulers for MTR WMNs

Scheduling link transmissions in WMNs is a challenging task. A schedule or a superframe of size T describes the set of links that are active in each slot, where T is the length of the schedule. The resulting schedule determines how efficiently the channel is being used; that is, it determines the capacity of a WMN and the end-to-end delays of flows. To determine whether a schedule is valid, in every time slot, all transmissions must be received successfully at the intended receiver. More specifically, an efficient schedule guarantees the maximal number of links is activated subject to the MTR constraint [16]; recall that this constraint ensures a node/router does not transmit *and* receive simultaneously. The following subsections outline single and multiple channels solutions that aim to maximize the number of activated links whilst minimizing the superframe length.

2.2.2.1 Single Channel

This section reviews approaches that operate over a single channel. The main reasons for employing a single channel are as follows: a) it is convenient to use a single channel for the backbone whilst other channels are used for local access, b) the more channels a network uses, the higher its operational cost because the IEEE 802.11b/a bands are licensed for outdoor use in some developing countries, and c) to avoid RF pollution as there are many WiFi networks in existence.

In [16], Raman et al. outlined a spatial TDMA MAC called 2 Phase (2P). The 2P protocol separates transmissions at each node into two phases: transmission and reception. In each time slot, for a given node, it switches either from transmission

phase (SynTx) to reception phase (SynRx) or vice-versa, while its neighbors switch from SynRx to SynTx or vice versa. SynTx and SynRx together are called SynOp. It is important to note that a node cannot be in both SynTx and SynRx concurrently. SynOp is maintained using a "token" mechanism. On a single link, a node A switches to SynTx until it receives the token from the other end of the link, say node B. After node A finishes sending data, it passes the token back to node B and switches to SynRx to receive data.

In [13], Chin proposes an algorithm to remove cliques from a topology. First, the algorithm finds the node with the largest degree. Then it examines the children of this node to see if any of them are connected. If so, the edges connecting these children are removed. The previous steps are then repeated on the node with the second largest degree until it processes all nodes. Finally, a standard graph coloring method is used on the resulting topology to obtain the final schedule. The disadvantage of this algorithm is that it leads to nodes using sub-optimal paths and may create bottleneck nodes. Interestingly, the resulting superframe length is much shorter than one that is derived using standard graph coloring on the original topology. Consequently, the average end-to-end delay between nodes is also shown to be lower.

To minimize superframe length but without extending the path length, a novel algorithm, called ALGO-1, is proposed by Chin et al. in [64]. ALGO-1 recursively partitions a WMN into maximally connected bipartite sets. ALGO-1 firstly partitions nodes into two disjoint sets: $Set1$ and $Set2$. This is done by checking a criterion associated with each link. For each node, say v, two values are introduced: $conn1$ is the number of nodes in $Set1$ that are connected to v, and $conn2$, a variable that corresponds to the number of nodes that are connected to v. The difference between these two values is denoted as δ, where $\delta = (conn2 - conn1)$. The node with the largest δ is picked and put into $Set1$, and the rest of the nodes are assigned into $Set2$. The aforementioned steps are repeated for nodes in $Set1$ and $Set2$ recursively until every node's δ is less than zero.

In [23], Chin et al. further improved ALGO-1 by applying a MAX-CUT algorithm to derive maximal bipartite graphs from a general network topology to realize the maximal number of concurrent transmitting links. They first define $Set1 = S$ and $Set2 = \bar{S}$ to be the MAX-CUT solution for a MTR WMN. In slot-1, each node in $Set1$ transmits to its neighbors in $Set2$; while in slot-2, nodes in $Set2$ transmit to its neighbors in $Set1$. When the first two slots finish, $Set1$ and $Set2$ are further divided into S and $\bar{S}$, respectively. The links established between the nodes of S and $\bar{S}$ within $Set1$ (or $Set2$) are called internal links. During time slot-3, the internal links originating from S to $\bar{S}$ of both $Set1$ and $Set2$ are activated. Lastly, in slot-4, the internal links from $\bar{S}$ to S are scheduled to operate. These steps are repeated until all links are activated at least once. In addition, to optimize the schedule, opportunistic links that are defined as links already scheduled in earlier slots, are added to subsequent slots. The way to add opportunistic links is to define a new graph G' by removing the vertices and edges that belong to $Set1$ and $Set2$ from $G(V, E)$. Then the algorithm is run again on graph G' to generate a pair of new disjoint sets $Set1'$ and $Set2'$. Finally, the algorithm adds the new sets into $Set1$ and $Set2$ respectively so that $Set1 = Set1 \cup Set1'$ and $Set2 = Set2 \cup Set2'$.

In [65] the network capacity is further improved using a MAX-CUT algorithm called ALGO-2, which is a modification of ALGO-1; see [23]. Using the same rules as ALGO-1, nodes are partitioned into $Set1$ and $Set2$ for transmitting and receiving, respectively. However, in [65], the authors divide nodes into $Set1$ and $Set2$ in every slot as opposed to every other slot. In addition, end-to-end delay is also considered. To minimize transmission delays without prolonging the superframe or impairing capacity, the authors proposed a Bucket Draining Algorithm (BDA) to reorder the slots in superframe S. To be specific, BDA first sorts links in descending order according to the weight of links or the number of time slots required by each link. Then slots that contain the heaviest weighted link in S are moved one by one into a new superframe called R until S is empty. By doing this, the activation of heavy links is more spread throughout the superframe, which creates fair and even activation of

links.

In [66], the authors propose a distributed version of the link scheduler in [65]. This scheduler is called Algo-d. It derives the shortest superframe that activates every link at least once while adhering to the MTR constraint. Similar to ALGO-1 [64] and ALGO-2 [65], nodes are divided into two sets S_1 and S_2 for transmitting and receiving, respectively. The key idea is to compute and update the value of $conn1$ and $conn2$ using only local information, where $conn1$ ($conn2$) is the number of nodes in S_1 (S_2) that a given node connects to. Then nodes use δ to determine its set membership, where δ is the difference between $conn1$ and $conn2$. However, data transmissions cannot commence until the final superframe is generated. In addition, there is no method to adapt the computed schedule when there is a topological change.

In [67], a weight w is assigned to each link in a MTR WMN. This weight reflects the traffic demand over a given link. The constraints in [67] are modified as follows: a link must be activated not once, but w slots in a given superframe, which guarantees that links are activated according to their actual traffic load. Chin et al. revised ALGO-1 [23] as follows. Firstly, they compute the sum weight of the cut $(S : \bar{S})$ using $w(S : \bar{S}) = \sum_{i \in S, j \in \bar{S}} w_{ij}$ of link e_{ij}. Then, all nodes in S are marked to transmit to the neighbor(s) in $\bar{S}$ in slot-1, and to receive from these neighbor(s) in slot-2. After that, ALGO-1 updates w_{ij} as follows: $w_{ij} = w_{ij} - 1$ if e_{ij} was used for Tx/Rx. If w_{ij} reaches zero, ALGO-1 removes e_{ij} from the network. These steps repeat until all w are equal to zero.

The authors of [15][54] and [68] outlined the following 2P limitation: the protocol requires all links in transmission or reception mode to be active for the same time duration regardless of the actual link traffic demand. This means nodes cannot adapt to varying traffic demands. This situation leads to two notable problems. The first one is inefficient throughput. Consider a link between node A and B. If traffic only flows from A to B, half of the link capacity is wasted, resulting in a lower throughput. The second one is unnecessary delay. For instance, when a

neighbor of a node wants to transmit to other nodes that do not interfere with existing transmissions, 2P will prohibit the transmission from happening, which leads to longer delays. These problems are addressed by [26] and [68].

In [26], Nedevschi et al. proposed an adaptive, distributed, high performance MAC called JazzyMAC to solve the fixed slot duration problem of 2P. JazzyMac exploits TDMA as well but its slot length is dynamic. A node arranges its slot size based on its changing traffic information. JazzyMAC works according to the following rules: (1) *token exchange*, there is one token for every link. When a node, say A, finishes its transmission to a neighbor, say B, it passes the token T_{AB} to node B. Attached is also a timeout value labeled as v_{AB} that corresponds to the time duration before node A is ready to receive data. Consequently, after node B receives token T_{AB}, it takes time v_{AB} for the token to be valid again, and to transmit to node A, (2) *mode*. A node can only switch from transmit to receive mode when it holds the token for all its links. Similarly, it can only transition from receive to transmit mode when it releases the token of all its links, (3) *transmission*, node A can start to transmit on link AB only when node A is in transmit mode and it holds a valid token T_{AB}. The authors showed that JazzyMAC improves the maximum network throughput in all considered scenarios by 15-100% as compared to 2P, especially in asymmetric traffic patterns because of better bandwidth allocation offered by dynamically sized slots.

Dai et al. [68] addressed the problem of deriving an optimum link schedule that satisfies traffic demands in a MTR WMN. The authors assume links have a weight ω corresponding to the number of slots required for transmission. They propose two schedulers called Heavy-Weight-First (HWF) and Max-Degree-First (MDF). The aim is to schedule links using the minimal total air-time to satisfy traffic demands. To obtain a schedule, initially, all the links from the conflict graph are sorted in descending order according to their weight. The algorithm selects non-interfering links one by one in descending order as per their weight to construct a Maximal Independent Set (MIS). To construct MIS, Dai et al. defined a maximal

matching as one set of links where no new links can be added into the set without causing interference to the links that are already in the set. After a matching E is determined, the scheduler identifies the link k with the minimum weight ω_k and assigns all links in the matching with ω_k time slots. Next, the scheduler removes link k from E and updates each link's weight by subtracting ω_k. Then, this process is repeated until all traffic demands in E are met. The steps of MDF are similar to HWF except that MDF sorts links according to their degree in the conflict graph.

In [27], Bao et al. propose a receive-oriented multiple access (ROMA) scheme in order to improve capacity. Their system uses Multi-Beam Adaptive Array (MBAA) antennas, meaning nodes are able to form several antenna beam-patterns simultaneously [69]. ROMA uses the neighbor-aware contention resolution (NCR) algorithm of [70], which requires two-hop topology information, to generate a schedule for each node in each time slot. It considers link weights, whereby a heavier weight corresponds to a higher probability of activation. Nodes are aware of their neighbors' ID and current time. These are then used to determine whether they are in transmit or receive mode in a pseudo-random manner. A node in receive mode then selects transmitting links. However, ROMA only provides four different weight values and no solution is presented on how they can be adjusted to achieve queue stability.

In [71], the authors present a distributed Reservation-based TDMA MAC protocol using Directional Antennas (RTDMA-DA) for use on a rectangular grid topology. Each node maintains a variable called *can_Res* that toggles between one and zero. Initially, the value of *can_Res* is set to one if the node's X and Y-coordinate are both even or both odd numbers. If *can_Res* has a value of one, a node transmits a Reservation Packet (RP) to its 1-hop neighbors. Otherwise, it receives from its neighbors. A frame consists of (i) one Reservation Frame (RF), which is used to reserve a slot for data transmission, and (ii) N Information Slots (ISs), which are used to transmit data packets. Here, the value of N is determined empirically. A node reserves an IS by transmitting a RP, which tells its neighbors that the corresponding IS is reserved for transmissions. To reduce bandwidth wastage, RTDMA-DA

allocates slots dynamically. Specifically, when a node has no packets to send, the unused bandwidth is passed to one of its neighbors who may have a higher traffic load. This protocol has many advantages. Firstly, it uses time slot reservation, which eliminates packet contention and collisions. As a result, the overall throughput when using RTDMA-DA is improved to at most 11 times better than IEEE 802.11 MAC. Secondly, RTDMA-DA considers fairness during its reservation procedure. This means each node is provided with an equal chance to transmit. Lastly, bandwidth utilization improves due to dynamic slot reservations.

In [72], Hazra et al. propose a measurement-based link scheduling for maritime WMNs that are deployed on buoys in port areas to serve ships. To maximize the number of transmissions in each time slot, the algorithm generates two sets: *assigned* and *not assigned*. Initially, all links are in the *not assigned* set. Then the algorithm traverses each link one by one in the *not assigned* set and schedules all links that do not interfere with previously scheduled links. If a link is scheduled into a time slot, it is moved to the *assigned* set. Otherwise, the link starts a counter called *skipped* to keep track of the number of consecutive slots that a link has been delayed. The algorithm preferentially selects links with a higher counter value. The algorithm terminates when all links are assigned a time slot.

The authors of [73] propose an efficient scheduling algorithm for WMNs that operate in the 60 GHz band. The network consists of a number of nodes called devices (DEVs); one of which is a central controller called PNC. Son et al.[73] assume that every node has up-to-date topological information. Each node is equipped with one DA that can steer its direction beam towards a particular node for data transmission/reception. The algorithm called Frame-based scheduling Directional MAC (FDMAC) that divides time slots into two phases: (i) a scheduling phase, which is for the PNC to poll DEVs for their traffic demand, compute a schedule S to satisfy the traffic demand, and send S to DEVs, (ii) a transmission phase, for DEVs to transmit data according to schedule S. To compute S, the PNC performs a greedy coloring algorithm on a directed and weighted multigraph, where

the weight represents the traffic demand of a link. Links are visited in decreasing order according to link weight.

In [74], Rhee et al. propose DRAND, a distributed randomized time slot scheduling algorithm to increase network capacity. This is the first distributed version of RAND [75], which is a centralized heuristic that generates efficient slot schedules for single channel, single radio WMNs. The authors assume conflict can occur among all nodes that are within two-hops away. The aim of DRAND is to schedule all links using the minimum number of time slots. It also takes into account the algorithm's running time and control overheads. It does not require time synchronization. DRAND operates as follows. Initially, a node A tosses a coin. If it gets a head, A runs a lottery that has a probability of $1/k$ to win, where k is the number of A's one-hop and two-hop neighbors that have not been scheduled a slot. After winning the lottery, A negotiates with its neighbors to select an interference-free time slot by sending transmission requests. Otherwise, node A returns to the initial state and tosses a coin again. Upon receiving a transmission request, a node B replies with a grant message containing B's available slots. After receiving the grant message from B, node A compares B's available slots to its own available slots and schedules a transmission from A to B in a common slot. Finally, node A sends a release message containing its busy slots to its neighbors. During the whole process, if A gets a tail, or losses the lottery, or no matching slot is found, node A returns to the initial state and tosses its coin again.

In [63], the authors proposed a concurrent transmission scheduling algorithm that maximizes the system throughput for mmWave networks. The system is an indoor IEEE 802.15.3c WPAN that consists of a number of wireless nodes and one piconet controller (PNC). Every node is equipped with a single electronically steerable DA so that transmitters and receivers can point their beams at each other for data transmission. In addition, the authors assume low user mobility and line-of-sight (LOS) transmission. One superframe is composed of three phases: a Beacon period (BP) for synchronization and for the PNC to disseminate control messages, a

contention access period (CAP) for nodes to send their specific throughput requirement to the PNC, and a channel time allocation period (CTAP) which contains at most M time slots for data transmission. To maximize the number of concurrent flows in every time slot while satisfying the throughput requirement of each flow, a flip-based algorithm is proposed. The main idea is to generate a link activation set U_i that contains the most links that can be activated simultaneously in each time slot i. To generate U_i, the PNC checks every link, and adds a link l to U_i if the transmission on l does not interfere with other links that are already in U_i. Once a link is selected to transmit in a particular slot, it remains active until its minimum throughput request is satisfied.

To further improve flow and network throughput, the technique in [76] breaks a one-hop transmission into multiple short-hop transmissions. The authors proposed a hop selection metric to select relays for data forwarding, and a multi-hop concurrent transmission (MHCT) scheme to exploit spatial reuse to allow non-interfering nodes to transmit simultaneously. To determine relay nodes, the PNC first calculates the associated weight W of each link based on its link length and traffic load. To help load-balance and improve flow throughput, the PNC runs Dijkstra's algorithm with associated weight W as the link cost. To maximize network throughput and improve network resource (time slots) utilization, the proposed MHCT scheme firstly sorts all link in decreasing link weight order. Then the approach presented in [63] is adopted to compute the link schedule.

In [19], Chu et al. use spatial multiplexing to improve transmission reliability and to increase data rate. The authors consider WMN where each node is equipped with a number of antenna elements. To maximize the number of concurrent streams in each time slot, the authors propose a scheduling scheme that chooses streams with high priority, where the value of priority depends on data type and delay. When scheduling streams in each time slot t, the algorithm visits each stream starting from the one with the highest priority and assigns it to slot t if the interference caused by this stream can be suppressed by additional antennas. The algorithm then considers

each stream in descending order in terms of its priority.

Chang et al. [77] propose a centralized link scheduling approach to maximize throughput whilst minimizing delay. They assume a cluster-based network topology that consists of a head node and a number of member nodes. Every node has a unique ID and is equipped with a k-beam smart antennas system. The authors also assume that the system can detect the beam forming direction of peer communicating nodes. The proposed algorithm consists of three phases. The first phase is *Request Collection*. In this phase, the head node first discovers its neighbors by sending polling messages on all its k beams. Then the head node collects its neighbors' transmission requirements including their volume of transmission data and their ID. In the second phase, called *Scheduling*, the algorithm constructs parallel transmission sets that contain the maximum number of links that can be activated simultaneously without interference. The final phase is used for data transmission. To explain the operation of phase 2, two kinds of parallel sets are used: Host-centric Parallel Set (HPS) and Cluster-centric Parallel Set (CPS). Firstly, the head node considers each member node M_1 and constructs a number of HPSs for M_1 that contains all parallel communications at different beams of M_1. To construct HPSs, the head node creates several empty HPSs and then greedily adds one link of the member node to an HPS if this link does not interfere with the links that already exist in the HPS. Secondly, the head node constructs a number of CPSs for the cluster, where each CPS is a collection of HPSs. To construct CPSs, the head node creates several empty CPSs and then greedily adds one HPS into an CPS whilst ensuring that no interfering links in the CPS. Finally, the head node assigns higher priority to CPSs with bigger size for transmission.

The authors in [78] and [79] jointly consider routing and scheduling in MTR WMNs. Specifically, in [78] each router is equipped with multiple DAs. The authors assume sufficient number of DAs to form links to all neighbors. All nodes operate over the same channel. They propose two solutions: JRS-Multi-DEC and JRS-BIP. Both aim to derive the superframe that has minimal length and yield low end-to-

end delay. To select an appropriate routing path, JRS-Multi-DEC first colors all links using the approach in [79]. Then for each flow, JRS-Multi-DEC selects the path with the lowest weight from the flow's k shortest paths. Here, the weight of a path is calculated by multiplying the length of a path, in terms of hops, by the superframe length. In another solution called JRS-BIP, the authors formulate the routing problem as a binary linear program. The objective is to minimize the number of hops of all traffic demands. Then both solutions use the algorithm in [65] to derive the link schedule.

In another work [79], Dutta et al. define the maximum concurrent flow problem as follows. Given an arbitrary network graph and a set of random source-destination traffic demands, find a feasible routing and its corresponding schedule that allows the maximum concurrent flow; this corresponds to the maximum fraction of all traffic demands that the routing and scheduling can satisfy. The authors formulated the problem as an exponential-sized Linear Program (LP) with flow conservation, and scheduling constraints. The objective is to ensure that the amount of flow on every link for every source-destination demand is maximized. The resulting LP problem has a very high computational complexity because as the number of nodes increases, the number of bipartite subgraphs increases exponentially. To decrease its complexity, an approximation algorithm is used. This algorithm is designed based on the following considerations: given a particular routing with a link flow vector $f(e)$, and the total capacity $C(e)$ of a link e, perform edge coloring using the DEC algorithm from [54]. Then find the corresponding routing that is able to achieve a utilization of at least $f(e)/C(e)$ for each link in the WMN.

2.2.2.2 Multi-Channel

Thus far, the aforementioned works have only considered a single channel. Under this assumption, the interference significantly arises with the increased number of nodes because all the nodes are sharing the same channel. In this section, all works consider multiple orthogonal channels. The key challenge is how to assign the limited

number of channels to nodes to ensure collision-free transmissions.

In [80], Chebrolu et al. designed a TDMA-based MAC, called wiFi-based Rural data ACess and TELephony (FRACTEL), that specifically targets WMNs deployed in rural regions of India; an example is the Ashwini network [81]. FRACTEL is designed to support voice and video or real-time applications, and supports both long-distance and local-access links. These links are formed using high-gain DAs and cover a distance of tens of kilometres. The DAs have a gain of 24-27 dBi with beamwidth of approximately 8°. The DAs are usually mounted on high-rise towers; specifically, towers that are 25 to 50 meter in height such that they satisfy line-of-sight and Fresnel clearance restrictions above trees and other obstructions. The authors defined a (ts_i, c_j) tuple, where ts_i is a time slot in the TDMA schedule, and c_j is a channel. The authors consider two fundamental problems: (1) time-slot allocation, aka TDMA scheduling, and (2) channel allocation, assuming three non-interfering channels; e.g., as in IEEE 802.11b/g. The link scheduling problem is to ensure that the same tuple is not allocated to two mutually interfering links, and to minimize the number of time slots used. This problem is reduced to the NP-hard node coloring problem. The researchers then propose a heuristic that considers two cases: (i) 1-hop links that start from the landline node, and (ii) 2-hop links, which connect nodes two hops away from the landline node. First, the heuristic colors 1-hop links. Given that all antennas at the landline node are placed on the same tower, all 1-hop links interfere with one another. Thus, the number of colors assigned to each 1-hop link equals the number of 1-hop links. Secondly, it considers 2-hop links. As the network topology from/to the landline node forms a tree, the 2-hop links coloring problem can be considered as a bipartite perfect matching problem. For every 2-hop link, it assigns a color for a 2-hop link that is the same as one of its non-interfering 1-hop link. The researchers did not consider 3-hop or more links because many practical scenarios involve only two hops links from the landline node.

In [82], Yu et al. propose a dynamic channel assignment and link scheduling algorithm to maximize capacity. They assume each node has multiple radios with

fast switching capabilities; each node is assumed to be able to switch to a different frequency within $80\mu s$. Thus a node can use a different channel in each time slot. The authors first construct a conflict graph to represent interference between links. Then the authors consider two types of applications: FTP and video. For FTP, they formulate a max-flow based ILP in order to achieve the maximal capacity whilst satisfying the radio and channel constraint under certain flow conservation constraints. As a result, all FTP flows are scheduled using the minimum number of time slots. For real-time video-type applications, a link-weight-adjusting strategy is employed to increase the minimal link satisfaction ratio, which is defined as the ratio of the flow rate to the required bandwidth among all links.

In [51], Ghosh et al. propose a centralized scheduling algorithm that takes both bandwidth and latency requirements into consideration. The authors assume a WiMAX or IEEE 8023.16 network [83]. A typical network model consists of one base station (BS) and several subscriber stations (SSs) that are connected to the BS located one or a small number of hops away. All stations are equipped with a single radio. The BS works as a central controller that determines the scheduling decisions, as well as the root node of the tree topology. The scheduling algorithm called Schedule Flow Subchannel (SFS) works in two steps. In step one, to satisfy delay requirement, BS calculates the farthest time slot x before a link is to be scheduled so that every flow's deadline is met. Then in step two, additional time slots before x can be chosen to satisfy the maximum bandwidth requirement. However, reference [51] does not consider the order in which consecutive links are scheduled. Specifically, for paths consisting of multiple links, i.e., L_1, L_2, L_3, if the slot for L_2 occurs earlier than the slot for L_1, the schedule spans at least two superframes. This increases end-to-end delays.

In references [80], [82] and [51], the authors solved the channel assignment and link scheduling problem for a tree topology. For arbitrary networks, Raman [84], as well as Dutta et al. [15] construct bipartite subgraphs, where each subgraph is located on a separate channel. Thus, as long as the channel subgraph is bipartite,

2P can be used independently within each subgraph.

In [84], Raman proposes a channel allocation method for MTR WMNs that uses the three non-overlapping channels of 802.11b/g. Raman considers the problem of asymmetric uplink/downlink capacity. The method gives network operators the freedom to divide a link's capacity in either direction based on actual traffic demands. For a link e, a network operator specifies a Desired Fraction (DF) f for one particular direction, and the remaining fraction $1-f$ for the opposite direction. Let $(V1, V2)$ be the bipartite graph of a channel. The edges in the direction from the set $V1$ to $V2$ are represented by $\overrightarrow{ei}$ and $\overleftarrow{ei}$ for the opposite direction. When 2P is applied to the subgraph, the actual Achieved Fraction (AF) for all $\overrightarrow{ei}$ is the same. The AF is termed as f', the value of which can be chosen freely. However, for any choice of f' in one subgraph, since the DF of all the $\overrightarrow{ei}$ is likely to be different in practice, we also have $AF \neq DF$. The difference between these two values, i.e., $|AF - DF|$, is defined as the mismatch of an edge. To completely remove mismatches, Raman presents Zero-Mismatch Channel Allocation (ZMCA), which groups edges into a number of different subgraphs such that all the DFs of the $\overrightarrow{ei}$ are the same.

A new channel allocation approach that is backward compatible with IEEE 802.11 is presented by Dutta et al. in [54]. They assigned only two non-interfering IEEE 802.11 channels to outgoing and incoming edges of a node respectively. The channel allocation problem is formulated as the minimum Directed Edge Coloring (DEC) problem. The goal is to find the minimum chromatic index γ, which is the number of colors required to color a graph. That is, for any vertex (node), the set of colors assigned to its incoming edges (links) is disjoint from that assigned to its outgoing edges. A simple solution is to carry out vertex coloring first, and then color all outgoing edges of a node with its assigned color. This means γ will be the same as the number of colors used in vertex coloring, say k. Instead of this naive method, Dutta et al. propose the following algorithm. Given a vertex-coloring of an undirected graph using k colors, they proved that the corresponding bidirectional graph can be edge colored with n colors, where n is the smallest integer that satisfies

$\binom{n}{\lfloor \frac{n}{2} \rfloor} \geq k$. With this new algorithm, γ is reduced from k to $2 \log k$, which indicates an exponential improvement.

Both references [62] and [15] consider joint link scheduling and routing problem for MTR WMNs. Specifically, Su et al. [62] propose a joint approach for multi-channel 60 GHz WMNs. Mesh routers have multiple radios and DAs. The authors solve the joint problem using a LP framework. Besides link flow, capacity, and half-duplex constraints, they also take into account (i) fairness, (ii) parallel transmissions, whereby a node can communicate with up to W neighbors concurrently, where W is the number of radios on a node, and (iii) the number of channels assigned to a node must be at most the number of available radios. To solve the LP and obtain a schedule that approximates the optimal network throughput, a greedy heuristic is proposed. First, the heuristic sorts unassigned flows in decreasing order in terms of traffic volume. Second, it assigns the first flow to the channel that can provide the maximum flow rate. After that, as links are half-duplex, it sets the capacity of this link and its reversed link to zero. The above procedures repeat until all flows are assigned a time slot and a channel. The results show that the proposed framework doubles, and in some cases more than four times the throughput of networks that use slotted Aloha. In another work [15], Dutta et al. adopted an iterative cut algorithm to generate K bipartite subgraphs for allocating K non-interfering IEEE 802.11 channels. To find a max-cut, the algorithm follows a simple local search. Firstly, the nodes are randomly separated into two sets. Each iteration of the algorithm is defined as a flip. It picks the node v that has more neighbors in its own set than that in the other set, and moves v to the other set. The algorithm terminates when no node can be flipped. As a result, the network graph is partitioned into a number of bipartite subgraphs. The authors assume that the capacity of all links in each bipartite subgraph from one set to the other set is the same.

2.3 Queue-Aware Link Scheduler

To date, a number of researchers have proposed queue length aware link scheduling methods for wireless networks where links have different and dynamic traffic load. In their seminal work, Tassiulas et al. [25] characterize the maximum attainable stability region in a multi-hop wireless network with random traffic arrivals. Specifically, this policy attempts to find a maximum weighted independent set of links in each slot, where the weight of links is determined from their queue length. The proposed throughput optimal scheduling policy solves a complex global optimization problem that stabilizes all queues whenever the offered traffic load vectors are in the interior of the stability region. However, under the k-hop interference model, the problem has been proven to be NP-Hard [85]. Thus, there remain significant implementation difficulties when the policy is extended to multi-hop networks due to its high complexity, especially in terms of signalling overheads. To achieve efficient implementation and low complexity, Tassiulas [86] presents a linear complexity centralized scheduling scheme. It can be viewed as a combination of the policy that schedules the set of links with the maximum aggregate backlog at each step, as in [25], plus a randomized link scheduling algorithm.

The scheme in [86] is then improved by Modiano et al. [12] where they propose a matching and gossiping algorithm that works in a distributed manner. The authors consider a multi-hop wireless network with a stochastic packet arrival process. The fundamental idea is that, in each slot, say slot i, the algorithm randomly selects an independent set of links, and compares it with the independent set scheduled in slot $i-1$. The one with the larger total weight wins and is scheduled in slot i, where the weight is the queue backlog for links. However, this policy requires semi-local information. This means that nodes need to collect queue length information from a few hops away, which requires many rounds of message exchanges.

In [11], the authors propose a Local Greedy Scheduler (LGS). They consider multi-hop wireless networks and aim to achieve the optimal stability region with a

low complexity scheduler. LGS is a distributed version of the centralized optimal scheduling policy called Greedy Maximal Scheduling (GMS) [87] or Longest Queue First (LQF) policy. The authors of [11] consider the k-hop interference model, in which a node can only communicate if none of its k-hop neighbors are transmitting. In the case of $k = 1$, the scheduler derives the maximal matchings whilst the $k = 2$ case is similar to the use of IEEE 802.11. LGS is a distributed protocol that selects links in decreasing order of queue length such that a link is activated if it has the locally longest queue length when comparing with its neighbors, and none of its interfering links are active. LGS approximates the performance of GMS.

Recently, a number of studies have focused on designing low-complexity and low-overhead distributed scheduling policies for multi-hop wireless networks. Although they are simple, the attained capacity region is typically smaller than their centralized counterparts [25]. Consequently, the authors of [88][89] and [90][91] defined the efficiency ratio as the largest γ such that for any supportable arrival rate λ achievable by the throughput optimal policy, this policy can support $\gamma\lambda$.

Chaporkar et al. [89] investigate the throughput performance under a class of distributed scheduling policies called maximal scheduling. The policy schedules a set S of links where every link in S has a packet to transmit and does not interfere with other links in S. The authors prove that the policy can be generalized to obtain a guaranteed fraction $\frac{1}{\beta}$ of the maximum stability region for arbitrary interference models, where β denotes the two-hop interference degree. Here, the interference degree of a link l is defined as the maximum number of links that interfere with link l and do not interfere with each other.

In [88], Lin et al. present a policy P for the node-exclusive interference model with constant computation time. Each link attempts to transmit based on a probability that is calculated from its queue length and the transmit/receive state of its one-hop neighboring links. The link with the smallest backoff time, which is set proportionally to its queue length, has a higher priority to transmit. The authors proved that the efficiency ratio γ of policy P is close to 1/3.

In [90], Gupta et al. modified policy P [88] and showed that for the one-hop interference model, this policy guarantees at most half of the capacity region. The authors assumed that the scheduling slot is further divided into M mini-slots. A link randomly picks an integer m from $1, 2, \ldots, M+1$ as its backoff time. If $1 \geq m \geq M$, similar with policy P, this link transmits at the m-th mini-slot unless it has already heard other transmissions. If $m = M+1$, it implies that this link will not attempt to transmit in this time slot. The main drawback of these two policies is that, if two or more links choose the same smallest backoff time, a collision will occur so that none of the links can transfer data in that time slot.

Bui et al. in [91] proposed a collision free distributed scheduling algorithm for multi-hop WMN. The aim is to control the tradeoff between throughput performance and control overhead with a parameter k. Specifically, for any integer $k \geq 1$, a link schedule will be generated using $4k+2$ round-trip time, where one round-trip time is the amount of time for a node to make a two-way handshake with its neighbors. This schedule is guaranteed to achieve a fraction of $\frac{k}{k+2}$ of the capacity region. To choose an appropriate k, the protocol designer needs to consider the physical condition of the network, such as mobility and arrival statistics. To generate a new schedule $\pi(t)$ which contains a set of non-interfering links to be activated at time slot t, the authors assume that $(q_t, \pi(t-1))$ is known, where q_t is the queue length at each link at slot t. The fundamental idea is to generate a new matching $\pi(t)$ based on the previous matching $\pi(t-1)$ and switch to the new matching if the gain is positive. Here, the gain is derived by subtracting the total weight of matching $\pi(t-1)$ from that of the new matching $\pi(t)$, where the weight is the queue length of each link. Thus, with every switch, the algorithm ensures that the total weight of activated links does not decrease.

Recently, a number of studies have focused on the problem of designing practical backpressure systems that are based on the throughput optimal backpressure algorithm of [25] in multi-hop WMNs. In [25], Tassiulas et al. introduced a max weight policy called backpressure scheduling that stabilizes all queues whenever the

offered traffic load vectors are in the interior of the capacity region. In particular, in each time slot, the policy calls for the activation of a set of non-interfering links with the maximum data rate and backpressure. Here, *backpressure* is defined as the difference in the queue size, also called the differential backlog, between transmitting and receiving ends of each link for each flow. This differential backlog is then used as the link weight. The work by Tassiulas et al. motivated a number of follow-up studies, including [92][93][94] and [95]. In [92], the authors propose a novel system design called Horizon to realize optimal routing and scheduling for WMNs where packets traverse several consecutive links. The aim is to schedule links and route packets efficiently and to guarantee fairness among users with long and short flows. This problem is defined by equation $r^* = \arg\max_r \sum_{i,j} \max_f (q_i^f(t) - q_j^f(t))$, which is used to define which links should be active and with what rate. Specifically, for every link (i, j), the authors first select the flow with the maximum backpressure $f(i, j) = \arg\max_f (q_i^f(t) - q_j^f(t))$. Then the appropriate rate vector r^* is selected to maximize throughput. However, Horizon can cause out of order delivery and is thus detrimental to TCP.

In [93], Warrier et al. propose a distributed solution that combines single path congestion control with backpressure based MAC scheduling for multi-hop wireless networks. The solution called DiffQ, is implemented between the transport and IP layer inside the Linux kernel, and supports a number of applications such as Web, Email and FTP using TCP or UDP. In DiffQ, every node i maintains a separate queue $Q_i(d)$ for each destination d. When there are packets need to be transmitted, DiffQ evaluates their destination queues and schedules the one with the largest backpressure. In addition, DiffQ has no restriction on traffic patterns. For congestion control, a node increases its rate when its queue decreases, and reduces its rate when the queue increases.

Laufer et al. [95] propose the first throughput-optimal architecture called XPRESS for wireless multi-hop networks. Similar to [93], it includes a congestion control scheme to ensure queue stability, and a multi-hop TDMA MAC protocol to perform

backpressure scheduling. The authors assume each node maintains a separate queue for each flow. They also assume that links are pre-assigned into a number of link sets, in which links can transmit in the same time slot without interference. Scheduling decisions are made on a slot-by-slot manner by a centralized backpressure scheduler as follows. First, the scheduler identifies the flow with the maximum differential queue backlog for each link. Then, the scheduler computes the total backpressure of each link set and chooses the link set with the maximum total backpressure for transmission. It, however, requires instantaneous per-flow queue backlog information, which incurs high signalling overheads.

2.4 Summary

This chapter has presented recent TDMA-based link scheduling approaches for WMNs that use directional antennas, different realizations of MTR WMNs, centralized and distributed link schedulers for MTR WMNs, and schedulers that consider queue lengths. Table 2.1 summarizes the link schedulers reviewed in this chapter. Current works have the following pros and cons:

1. Although there are a number of TDMA-based MACs, they assume the node-exclusive spectrum sharing model; critically, each node can only communicate with at most one other node at any point in time. However, the node-exclusive model is not representative of MTR wireless networks. Thus, this thesis focuses on link schedulers that can be applied to MTR WMNs where multiple concurrent transmissions or receptions take place simultaneously at each node.

2. A number of centralized link schedulers for MTR WMNs are proposed to maximize capacity[16] [14] [54] [13] [64] [23] [65]. However, they do not consider links with different transmission duration. In practice, different air-time is necessary because links are likely to have different loads and data rates. Only the algorithms in [67] [68] and [62] addressed this problem. However, these

works require a group of links to start transmission at the same time. To this end, Chapter 3 proposes the first centralized scheduler that allows individual transmissions to start as soon as they do not cause interference, and finish whenever their air-time runs up.

3. Centralized TDMA MACs such as [72][73][63][76][19][77][78][79][80][82][51][84] and [15] are not practical for large-scale MTR WMNs as they require a central controller node to compute and disseminated the schedule. Consequently, a large proportion of capacity is wasted on propagation of control overheads rather than data transmissions. To date, there are only a handful of distributed schedulers. Reference [26] is semi-distributed because it requires an initial token assignment which is done centrally. The distributed approaches in [66][27][71] and [74] do not provide a method to adapt the computed schedule when there is a topological change. These limitations motivate the design of a distributed scheduler presented in Chapter 4 that solves the above problems.

4. For WMNs where queue length of each link is known as a link weight, the centralized approach in [25] and the semi-distributed approach in [12] are throughput optimal. However, they require nodes to collect information from multiple hops away and thus incur an enormous overhead. Researchers thus sought distributed policies to minimize computational complexity and signaling overheads; see [11][89][88][90] and [91]. Although these distributed policies are simple, their stability region reaches at most half of the theoretical maximum. In order to increase their stability region, a number of works [92] [93] and [95] use backpressure approaches. However, these works do not consider nodes with MTR-capability. Indeed, only reference [27] has considered queue lengths for MTR WMNs. However, it is unclear whether the algorithm ensures queue stability and its performance in multi-hop flow scenarios is unknown. Hence, Chapter 5 presents the first distributed scheduler that addresses the aforementioned issues.

Scheduler	**Centralized or Distributed**	**Channel**	**Radio**	**Hop**	**MTR**	**JRS**	**Weight**	**Topology**	**Aim**
[48]	Centralized	Single	Multi	Single	No	No	No	Arbitrary	QoS
[39]	Distributed	Single	Single	Single	No	No	No	Arbitrary	Max throughput
[34]	Distributed	Single	Single	Single	No	No	Yes	Cluster	Max throughput/ Min delay
[37]	Centralized	Single	Single	Single	No	No	No	Arbitrary	Max throughput/Min power
[50]	Centralized	Single	Single	Single	No	No	Yes	Tree	Max capacity
[30]	Centralized	Single	Single	Multi	No	No	Yes	Sinktree	Min delay
[31]	Centralized	Single	Single	Multi	No	Yes	No	Arbitrary	Min superframe
[40]	Centralized	Single	Single	Multi	No	Yes	Yes	Arbitrary	Min superframe
[29]	Centralized	Single	Single	Multi	No	Yes	Yes	Grid	Min delay
[52]	Centralized	Multi	Multi	Multi	No	Yes	No	Arbitrary	Max throughput
[16]	Centralized	Single	Multi	Single	Yes	No	No	Bipartite	Max capacity
[14]	Centralized	Multi	Multi	Single	Yes	No	No	7-node	Max capacity
[54]	Centralized	Multi	Multi	Single	Yes	No	No	Arbitrary	Max concurrent flows
[59]	Centralized	Multi	Multi	Single	Yes	No	No	Arbitrary	Max throughput
[13]	Centralized	Single	Multi	Single	Yes	No	No	Arbitrary	Max capacity

[64]	Centralized	Single	Multi	Single	Yes	No	No	Arbitrary	Max capacity
[23]	Centralized	Single	Multi	Single	Yes	No	No	Arbitrary	Max concurrent links
[65]	Centralized	Single	Multi	Single	Yes	No	No	Arbitrary	Max capacity
[66]	Distributed	Single	Multi	Single	Yes	No	No	Arbitrary	Max capacity
[67]	Centralized	Single	Multi	Single	Yes	No	Yes	Arbitrary	Max capacity
[26]	Distributed(semi)	Single	Multi	Single	Yes	No	Yes	Arbitrary	Max throughput
[68]	Centralized	Single	Multi	Single	Yes	No	Yes	Arbitrary	Max capacity
[27]	Distributed	Single	Multi	Single	Yes	No	Yes	Arbitrary	Max capacity
[71]	Distributed	Single	Multi	Single	Yes	No	No	Grid	Max throughput
[72]	Centralized	Single	Multi	Single	Yes	No	No	Arbitrary	Max capacity
[73]	Centralized	Single	Single	Single	Yes	No	No	Arbitrary	Max throughput/Min delay
[74]	Distributed	Single	Single	Single	Yes	No	No	Arbitrary	Max capacity
[63]	Centralized	Single	Single	Multi	Yes	No	No	Arbitrary	Max throughput
[76]	Centralized	Single	Single	Multi	Yes	No	No	Arbitrary	Balance traffic load
[19]	Centralized	Single	Multi	Single	Yes	No	No	Arbitrary	Max capacity
[77]	Centralized	Single	Multi	Single	Yes	No	No	Cluster	Max throughput/Min delay

[78]	Centralized	Single	Multi	Multi	Yes	Yes	No	Arbitrary	Max concurrent flow
[79]	Centralized	Single	Multi	Multi	Yes	Yes	No	Arbitrary	Max concurrent flow
[80]	Centralized	Multi	Multi	Multi	Yes	No	No	Tree	QoS
[82]	Centralized	Multi	Multi	Single	Yes	No	No	Tree	Max capacity
[51]	Centralized	Multi	Multi	Multi	Yes	No	No	Tree	QoS
[84]	Centralized	Multi	Multi	Single	Yes	No	No	Arbitrary	Bandwidth guarantee
[62]	Centralized	Multi	Multi	Multi	Yes	Yes	Yes	Arbitrary	Max capacity/fairness
[15]	Centralized	Multi	Multi	Single	Yes	Yes	No	Arbitrary	Max capacity
[12]	Distributed(semi)	Single	Single	Multi	No	No	Yes	Arbitrary	Max throughput
[11]	Distributed	Single	Single	Multi	No	No	Yes	Arbitrary	Capacity overhead tradeoff
[89]	Distributed	Single	Single	Single	No	No	Yes	Arbitrary	Max capacity
[88]	Distributed	Single	Single	Single	No	No	Yes	Arbitrary	Throughput complexity tradeoff
[90]	Distributed	Single	Single	Multi	No	No	Yes	Arbitrary	Throughput complexity tradeoff

[91]	Distributed	Single	Single	Multi	No	No	Yes	Arbitrary	Throughput complexity tradeoff
[25]	Centralized	Single	Single	Single	No	No	Yes	Arbitrary	Obtain max stability region
[92]	Distributed	Single	Single	Multi	No	Yes	Yes	Arbitrary	Max capacity/fairness
[93]	Distributed	Single	Single	Multi	No	No	Yes	Arbitrary	Max capacity
[95]	Centralized	Single	Single	Multi	No	No	Yes	Arbitrary	Max capacity

Table 2.1: Comparison of works reviewed in Chapter 2

Chapter 3

Scheduling Links with Air-Time

As discussed in Chapter 2, a number of researchers have proposed TDMA-based link schedulers for MTR WMNs. The fundamental problem is to determine the shortest superframe length that ensures all links are activated at least once. In other words, the aim is to minimize the time in which the last link finishes transmission. However, current solutions require all links in transmission or reception mode to be active for the same time duration regardless of the actual link load. As a result, this situation leads to capacity waste and increases delay. To this end, this chapter proposes a new algorithm called A-TxRx that addresses the aforementioned issue. In particular, A-TxRx addresses the following problem: given a MTR WMN, whereby each link has a different air-time, design a centralized algorithm that derives the minimal superframe length whereby links are activated at least once and in accordance to their requested air-time.

In a nutshell, this chapter makes the following contributions:

- It establishes the network model and presents some preliminary analysis on the link scheduling problem at hand. In particular, it shows the superframe bound for bipartite, linear, ring, grid, fully connected, and arbitrarily connected topologies.

- It proposes A-TxRx, which is the first centralized link scheduler that activates

links with different transmission duration or *air-time* in a MTR WMN. Unlike existing schedulers, A-TxRx aims to minimize the finishing time of the last transmitting link as well as maximize the number of links at any time instant. According to simulation results, the superframe length of A-TxRx reaches at most 70% shorter than state-of-the-art approaches.

- It outlines an improvement to A-TxRx, where the algorithm opportunistically adds scheduled links to further increase network capacity. As a result, the number of concurrent transmitting links of A-TxRx is about 40% more than that of JazzyMAC [26] on average.

- It computes the running time of A-TxRx, proves that the algorithm derives a collision-free schedule for arbitrary topologies. In addition, it analyzes the bound of superframe length for bipartite topologies when running A-TxRx.

The rest of this chapter is structured as follows. Section 3.1 describes the network model of MTR WMNs. Then Section 3.2 identifies the problem and provides some preliminary analysis. The details of A-TxRx are presented in Section 3.3. The properties of A-TxRx are discussed in Section 3.4. The research methodology is presented in Section 3.5. Section 3.6 presents experiments and results. Then this chapter concludes in Section 3.7.

3.1 Network Model

A MTR WMN is modeled as a directed graph $G(V, E)$. The set V is comprised of nodes that are equipped with $b \geq 1$ directional antennas. Each node u has a transmission range of r and $b_u \geq 1$ radios. There is a directional edge $(u, v) \in E$ connecting node u and v if they are within each other's transmission range. Also, consider there to be a directional link connecting node v to u; i.e., $(v, u) \in E$. Note, in practice, nodes may have a different transmission range due to the vagaries of the wireless channel. To this end, nodes are required to ensure incoming and outgoing

link to each neighbor is functional. This can be achieved via HELLO messages in the neighbor discovery process whereby nodes include the neighbors they can hear in their HELLO messages. The function $f_t : E \to \mathbb{R}$ returns the allocated air-time of a given link. Hence, f_t models the required transmission time in order to meet a given traffic load. All nodes are able to concurrently transmit or receive on all links. Each link is supported by a radio and assume $b_u \geq |N(u)|$ for all nodes. Formally, there is the following *Mix-Tx-Rx constraint:* for a given node u, let $IN(u,t)$ and $OUT(u,t)$ be its set of receiving and transmitting links at time t respectively. This constraint is met if both $|IN(u,t)|$ and $|OUT(u,t)|$ are *not* greater than zero simultaneously at any time t for all node $u \in V$.

In the model, the following assumptions are made:

- All nodes are synchronized globally. This is reasonable given that nodes are static and can use GPS to synchronize their clock.
- Nodes are tuned to a single frequency.
- Assume the traffic load on each link is aggregated, i.e., see [26], and remains fixed for a non-negligible amount of time, e.g., every hour. This also means the routing for source destination pairs is fixed for this period of time.
- Each air-time unit can correspond to a specific amount of time, e.g., one second, or the time it takes to transmit one packet. It is important to note that once an air-time is assigned to a link, upon scheduled for transmission, it is guaranteed to transmit for said air-time without pre-emption.

3.2 The Problem

The problem at hand is as follows: given a MTR WMN, whereby each link has a different air-time, design a centralized algorithm that derives the minimal superframe length whereby links are activated at least once and in accordance to their requested air-time. Note, a superframe is defined as the sequence $(\{e_1, \cdots e_x\}, t_1) \cup$

$\ldots \cup (\{e_y, \cdots e_{|E|}\}, t_{|E|})$, whereby $e_i \in E$ is the link to be scheduled at time t_i. For example, in Figure 1.3, the superframe is $(\{AB, AC\}, 0) \cup (\{BA, CA\}, 10) \cup (\{BC\}, 15) \cup (\{CB\}, 24)$. The problem is to derive a superframe with minimal $t_{|E|} + f_t(e_{|E|})$ value subject to the Mix-Tx-Rx constraint (cf. Figure 1.1), and each link $e \in E$ receiving at least $f_t(e)$ of air-time.

Note, if all links have the same air-time, then the problem is similar to prior works such as [23]. In particular, the authors of [23] show that deriving the maximum number of links in each slot is equivalent to solving the NP-complete, MAX-CUT problem, meaning the problem at hand is just as hard. To this end, the next section proposes a greedy heuristic that aims to determine the shortest possible superframe.

3.2.1 Preliminary Analysis

Before outlining A-TxRx, the superframe length bound of a few example topologies is presented. Let $\delta(G)$ be the superframe length of the link schedule for a graph $G(V, E)$.

3.2.1.1 Example I: Bipartite topology

Recall that nodes of a bipartite graph $G(V, E)$ can be divided into two sets $\mathbb{A}$ and $\mathbb{B}$. Further, by definition, each node in $\mathbb{A}$ and $\mathbb{B}$ has no incoming or outgoing link to any other node in $\mathbb{A}$ and $\mathbb{B}$ respectively. Let $l_t^{\mathbb{A}}$ and $l_t^{\mathbb{B}}$ be the longest air-time of links emanating from nodes in set $\mathbb{A}$ and $\mathbb{B}$ respectively.

Lemma 1. *The superframe length of the link schedule for a bipartite graph $G(V, E)$ is $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$.*

Proof. Without loss of generality, in slot 1, let each node in set $\mathbb{A}$ transmit to its neighbors in set $\mathbb{B}$ for $l_t^{\mathbb{A}}$ time, and in slot 2, each node in set B transmit to its neighbors in set $\mathbb{A}$ for $l_t^{\mathbb{B}}$ time. Thus, the superframe length of the 2-slot schedule is $l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$ because the air-time of each link emanating from any node in $\mathbb{A}$ ($\mathbb{B}$) is at most $l_t^{\mathbb{A}}$ ($l_t^{\mathbb{B}}$). □

Using Lemma 1, the following corollaries are proposed.

Corollary 1. *The link schedule for an arbitrary connected graph $G(V, E)$ that can be partitioned into k bipartite graphs has an upper bound superframe length of $\delta(G) = \sum_{i=1}^{k} l_i^{\mathbb{A}} + l_i^{\mathbb{B}}$, where $l_i^{\mathbb{A}}$ ($l_i^{\mathbb{B}}$) is the longest air-time of links emanating from nodes in set $\mathbb{A}(\mathbb{B})$ of each bipartite graph G_i, for $i = 1, 2, \cdots, k$.*

Proof. Consider a bipartite graph G_i among the k bipartite graphs. Using Lemma 1, $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$. Without loss of generality, let the schedule for G starts from links in G_1 that require $\delta(G_1) = l_1^{\mathbb{A}} + l_1^{\mathbb{B}}$, followed by links in G_2 with $\delta(G_2) = l_2^{\mathbb{A}} + l_2^{\mathbb{B}}$, and so on to links in G_k that use $\delta(G_k) = l_k^{\mathbb{A}} + l_k^{\mathbb{B}}$. Thus the schedule has a superframe length of $\delta(G) = \sum_{i=1}^{k} l_i^{\mathbb{A}} + l_i^{\mathbb{B}}$. □

Corollary 2. *The link schedule for an arbitrary connected graph $G(V, E)$ has an upper bound superframe length of $\delta(G) = \sum_{i=1}^{2\times\lceil\log_2|V|\rceil} l_i^{\mathbb{A}} + l_i^{\mathbb{B}}$, where $l_i^{\mathbb{A}}$ ($l_i^{\mathbb{B}}$) is the longest air-time of links emanating from nodes in set $\mathbb{A}(\mathbb{B})$ of each bipartite graph G_i, for $i = 1, 2, \cdots, 2 \times \lceil\log_2|V|\rceil$.*

Proof. The first thing to do is to prove the bound for a fully connected graph $J(V', E')$. Then, the result of J is used to compute the bound for any arbitrary connected graph $G(V, E)$, for $V = V'$ and $E \subseteq E'$. Theorem 7.1 in [23] shows that the link schedule for the network with $|V| > 1$ nodes has a superframe length of $2 \times \lceil\log_2|V'|\rceil$. In other words, $J(V', E')$ can be partitioned into $2 \times \lceil\log_2|V'|\rceil$ bipartite graphs since the nodes in each slot of the superframe form a bipartite graph G_i. Then compute the bound $\delta(J) = \sum_{i=1}^{2\times\lceil\log_2|V'|\rceil} l_i^{\mathbb{A}} + l_i^{\mathbb{B}}$ for $J(V', E')$ using Corollary 1 for $k = 2 \times \lceil\log_2|V'|\rceil$. Given an arbitrary graph $G(V, E)$, for $V = V'$ and $E \subseteq E'$, one can always construct a fully connected graph $J(V', E')$ from G by including a set of links E'' that $E'' = E' \setminus E$. Further, setting the air-time of each link in E'' to zero, one can also produce the link schedule for G from the schedule for its corresponding J by ignoring all links in E''. Specifically, such schedule for G has the same superframe length as that for its J, i.e., $\delta(G) = \sum_{i=1}^{2\times\lceil\log_2|V|\rceil} l_i^{\mathbb{A}} + l_i^{\mathbb{B}}$. □

Notice that in the foregoing proof the superframe length is an upper bound because in the two-slot link schedule of G_i, it may be possible to include links from other bipartite graphs, as long as the addition does not violate the Mix-Tx-Rx constraint. Specifically, consider a link emanating from a node in set $\mathbb{A}$ of a bipartite graph G_i that has air-time t, where $t < l_i^{\mathbb{A}}$. Upon completion, there may be links in a bipartite graph G_j, for $i \neq j$, with duration $(l_i^{\mathbb{A}} - t)$ that do not conflict with any links in G_i. These so called opportunistic links can thus be added to the schedule for G_i, which may help reduce $\delta(G)$. As we will see later in Section 3.3.1, exploiting opportunistic links also leads to higher network capacity.

3.2.1.2 Example II: Linear topology

Each node in a linear topology $G(V, E)$ only communicates with its one-hop neighbors using half-duplex links. When all links in G have the same air-time, $l_t(e)$, its link schedule has a superframe length of $\delta(G) = 2l_t(e)$. To see this, number all nodes in G sequentially from left to right, starting from 1 to $|V|$. In the first slot, all the odd numbered nodes are scheduled to transmit to the even numbered nodes for $l_t(e)$ time unit(s). In the second slot, the even numbered nodes become transmitters and the odd numbered nodes become receivers for another $l_t(e)$ time unit(s). As a result, the superframe length is $\delta(G) = 2l_t(e)$.

When the air-time of the links is different, the links are no longer limited by one specific time slot. Instead, they are activated continuously subject to the Mix-Tx-Rx constraint. Consider the linear topology of Figure 3.1; the label on each link shows its air-time. Link e_{21} and e_{23} are in conflict with link e_{12} and e_{32}. Henceforth, link e_{21} and e_{23} can operate in the same slot for a duration of $MAX(3, 2) = 3$, and link e_{12} and e_{32} must operate in another slot for a duration of $MAX(3, 1) = 3$. Thus, the schedule for the four incident links to node 2 takes $3 + 3 = 6$ time units. With this observation, the following Lemma 2 that gives the superframe length of link schedule for the linear topology $G(V, E)$ is proposed.

Let $IN(u)$ be the set of all incoming links to node u, $OUT(u)$ be the set of all

outgoing links from node u, $l_t(IN(u))$ be the longest air-time among incoming links to node u, and $l_t(OUT(u))$ be the longest air-time among all outgoing links to node u. For a linear topology, $IN(u) = OUT(u) = 2$, except for $u = 1$ or $u = |V|$, in which case $IN(u) = OUT(u) = 1$. For the topology in Figure 3.1, $IN(2) = \{e_{12}, e_{32}\}$, $OUT(2) = \{e_{21}, e_{23}\}$, $l_t(IN(2)) = MAX(3, 1) = 3$ and $l_t(OUT(2)) = MAX(3, 2) = 3$.

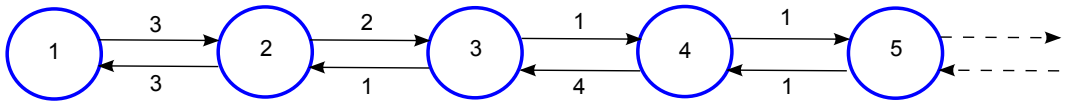


Figure 3.1: A linear topology

Lemma 2. *The superframe length of the link schedule for a linear topology $G(V, E)$ has an upper bound of $\delta(G) = MAX(l_t(OUT(u))) + MAX(l_t(IN(u)))$, for $u \in V$.*

Proof. A linear topology is a bipartite graph and therefore its nodes can be divided into two sets $\mathbb{A}$ and $\mathbb{B}$. Without loss of generality, let set $\mathbb{A}$ and $\mathbb{B}$ contain all odd numbered and all even numbered nodes in V, respectively. Further, let each node in $\mathbb{A}$ transmit at slot 1 and each node in $\mathbb{B}$ at slot 2. For all odd numbered nodes $u \in V$, at slot 1, each node u in set $\mathbb{A}$ transmits to its even numbered neighbors in set $\mathbb{B}$ for $MAX(l_t(OUT(u))$ time unit(s) and, at slot 2, each node u receives from its neighbors for $MAX(l_t(IN(u)))$ time unit(s). Thus, the superframe length is $\delta(G) = MAX(l_t(OUT(u))) + MAX(l_t(IN(u)))$. Similarly, for odd numbered nodes $u \in V$, one will obtain the same schedule with the same superframe length. □

As an example, for Figure 3.1 using odd numbered nodes $u \in V$, i.e., $u \in \{1, 3, 5\}$, the superframe length is $MAX(3, 1, 1) + MAX(3, 4, 1) = 7$. Similarly, using even numbered nodes $u \in V$, i.e., $u \in \{2, 4\}$, one obtains the same superframe with length $MAX(3, 4) + MAX(3, 1) = 7$.

3.2.1.3 Example III: Ring topology

Similar to the linear topology, each node in a ring topology $G(V, E)$ only communicates with its one-hop neighbors using half-duplex links. Let us number all nodes in

G sequentially from left to right, starting from 1 to $|V|$. Unlike a linear topology, the node 1 in a ring topology has one outgoing link and one incoming link to and from the node $|V|$ respectively. One can convert a ring topology $G(V, E)$ into a linear topology $L(V, R)$ that contains the same number of nodes $|V|$ by removing one pair of links from E, i.e., $|R| = |E| - 2$. For any node $u \in V$, $u \neq 1$ and $u \neq |V|$, the removed links can be a pair $\{e_{u(u+1)}, e_{(u+1)u}\}$ or another pair $\{e_{u(u-1)}, e_{(u-1)u}\}$; for $u = 1$, node $(u - 1)$ is $|V|$, while for $u = |V|$, node $(u + 1)$ is 1. Let us denote each such pair of links for node u as $\{e_{u*}, e_{*u}\}$. Notice that a ring topology with an even number of nodes $|V|$ is a bipartite graph, eg., Figure 3.2, but the topology with an odd number of nodes $|V|$ is not a bipartite graph.

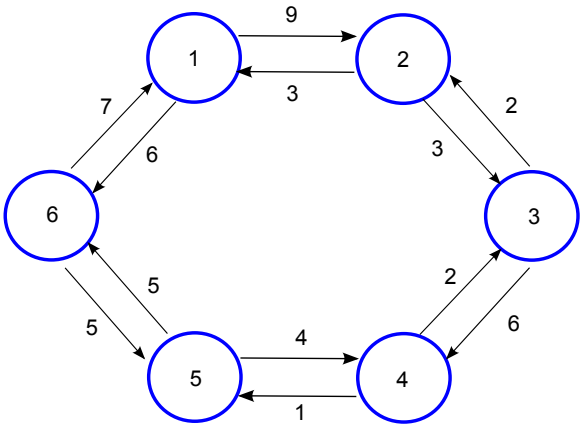


Figure 3.2: A ring topology consists of even number of nodes

Lemma 3. *The superframe length for a ring topology $G(V, E)$ is upper bounded by (i) $MAX(l_t(OUT(u))) + MAX(l_t(IN(u)))$, for even number $|V|$, and (ii) $MAX(l_t(OUT(u))) + MAX(l_t(IN(u))) + MIN(l_t(e_{v*}) + l_t(e_{*v}))$, for odd number $|V|$.*

Proof. Since a ring topology $G(V, E)$ with even number of nodes $|V|$ is a bipartite graph, for case (i), one can directly apply Lemma 2 to obtain the superframe length. For case (ii), first obtain a pair of links $\{e_{v*}, e_{*v}\}$ that has the minimum $l_t(e_{v*})+l_t(e_{*v})$, i.e., $MIN(l_t(e_{v*})+l_t(e_{*v}))$. After removing the said pair of links from $G(V, E)$,one can obtain a linear topology $L(V, R)$, and use Lemma 2 to compute its superframe length, i.e., $MAX(l_t(OUT(u))) + MAX(l_t(IN(u)))$. Taking the sum of

the air-time of the chosen pair of links and the superframe length of $L(V, R)$, the duration for case (ii) is obtained. □

The ring topology in Figure 3.2 has six nodes. For odd numbered nodes $u \in \{1, 3, 5\}$, the superframe length is $MAX(9, 6, 5)+MAX(7, 3, 5) = 16$. For even numbered nodes $u \in \{2, 4, 6\}$, the same superframe length is obtained using Lemma 3: $MAX(3, 2, 7) + MAX(9, 6, 6) = 16$.

3.2.1.4 Example IV: Grid topology

An $m \times n$ grid consists of m rows and n columns of nodes. Each node on the x-th row and y-th column in the grid is denoted by its coordinate (x, y). All nodes have exactly four neighbors, except for those located at the grid boundaries, i.e., those whose coordinates have either $x = 1$, $x = m$, $y = 1$ or $y = n$. Further, except for the four corners, i.e., $(1, 1)$, $(1, n)$, $(m, 1)$ and (m, n), boundary nodes have three neighbors. Figure 3.3 shows a 3×3 grid. The grid topology is bipartite. To see this, one can divide its nodes into two groups: (i) Group-1 contains nodes with coordinate $x \ \% \ 2 = y \ \% \ 2$; (ii) Group-2 contains nodes with coordinate $x \ \% \ 2 \neq y \ \% \ 2$. In Figure 3.3, filled circles are Group-1 nodes; otherwise, they belong to Group-2. Notice that each Group-1 node has neighbors only in Group-2, and vice versa, and thus the grid is bipartite. For a grid topology $G(V, E)$, let V_1 (V_2) denote the set of nodes in Group-1 (Group-2); thus $V = V_1 \cup V_2$, and $|V| = |V_1| + |V_2|$ because $V_1 \cap V_2 = \emptyset$.

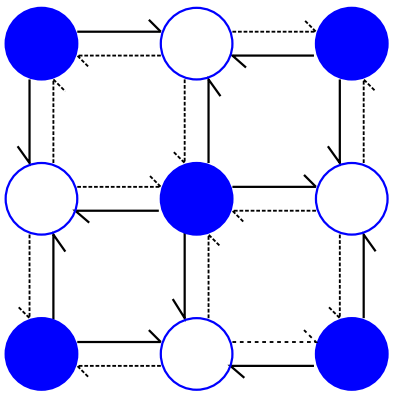


Figure 3.3: A grid topology

Lemma 4. *The superframe length of a grid topology $G(V, E)$ is upper bounded by $MAX(l_t(OUT(u))) + MAX(l_t(OUT(v)))$, for $u \in V_1$ and $v \in V_2$.*

Proof. Given that grid G is bipartite, one can set $\mathbb{A} = V_1$ and $\mathbb{B} = V_2$, and use Lemma 1 to obtain the superframe length. □

Let us assume an equal air-time of 5 for every link in Figure 3.3, according to Lemma 4, the superframe length is $5 + 5 = 10$.

3.3 The Solution: A-TxRx

The basic idea is to greedily schedule new non-interfering links whenever a link finishes transmission. Section 3.3.1 shows how the resulting schedule can be improved by adding so called opportunistic links. In addition, Section 3.3.2 simplifies a computationally expensive step used to calculate a MIS with a greedy step that adds non-conflicting links according to their transmission time.

A-TxRx has the following key steps. Firstly, it constructs a conflict graph $G'(V', E')$ based on the network topology $G(V, E)$. In the conflict graph, each vertex $v' \in V'$ denotes a link in E, and a conflict between two links in E is represented by an edge $e' \in E'$ between the corresponding vertices. It then greedily determines all Maximal Independent Sets (MISs) of G'. Recall that an MIS is the subset of all the links in G' that can be activated at the same time without interference. Note that a link may appear in different MISs. In the third step, the MIS with the most links is chosen, which ensures high throughput. The algorithm activates all links in the selected MIS, and label them as *active links*. Their air-times are also recorded. Among all the *active links*, the ones that have the same minimum air-time are regarded as *finishing links*. These finishing links' current air-time is then recorded. At such time, remove them from G'. If G' is not empty, A-TxRx first determines if there are any active links. These links are denoted as *remaining links*. If there are no *remaining links*, then a new MIS is obtained directly from the current G'. This

MIS contains new links that can be added into the superframe. However, if there are *remaining links*, check to see whether new non-interfering links can be added. A-TxRx first constructs a new conflict graph G'' from G'. Specifically, it removes the finishing links, all remaining links and their neighbors. Then a new MIS is obtained from G'', which contains the most new links that do not interfere with remaining links. The next set of *finishing links* are then determined and the algorithm repeats the process. A-TxRx terminates when G' is empty.

Symbol	*Description*
G	The directed graph
V	The set of nodes or vertices in G
E	The set of links or edges in G
G'	The conflict graph generated from G
V'	The set of nodes or vertices in G'
E'	The set of links or edges in G'
G''	The altered graph from G'
e_{AB}	A link with source node A and destination node B in E
$\delta(G)$	The maximum length of the superframe derived by A-TxRx for graph G
$f_t(e)$	The air-time of link e
$V_1 \backslash V_2$	The pair of subsets of a bipartite graph
$i_1 \backslash i_2$	The link emanating from V_1 or V_2 with the longest air-time
$\mathbb{A} \backslash \mathbb{B}$	Two MISs derived from the conflict graph of a bipartite graph
$l_t^{\mathbb{A}} \backslash l_t^{\mathbb{B}}$	The activation time of links in $\mathbb{A}$ or $\mathbb{B}$
$\mathcal{A}$	The set of active links
$\mathcal{F}$	The set of finishing links
$\mathcal{R}$	The set of remaining links
$\mathcal{N}$	The set of interfering nodes of links in $\mathcal{R}$
t_{finish}	The next finishing time of the currently active links

Table 3.1: Key notations in Chapter 3

Algorithm 1 shows how A-TxRx determines the schedule for the topology shown in Figure 1.3. As depicted, it is an MTR WMN with three nodes A, B, and C connected with bidirectional links. The value next to each link indicates its allocated air-time.

- A-TxRx takes as input the network graph $G(V, E)$, and the air-time $f_t(e)$ assigned to each edge $e \in E$.
- A-TxRx produces SF as its output. The set SF contains tuples $(\mathcal{A}, t)$, where

Algorithm 1: A-TxRx

Input: network graph $G(V, E)$, air-time of links $f_t : E \to \mathbb{R}$
Output: SF containing set of links $\mathcal{A}$ and their activation time t

```
1  if |V| ≤ 1 then
2      return
3  else
4      t ← 0
5      SF ← ∅
6      A = F = R ← ∅
7      G' ← ConflictG(G(V, E))
8      while (G' ≠ ∅) do
9          if (R = ∅) then
10             A ← MaxMIS(G')
11             SF ← SF ∪ (A, t)
12             G' ← G' − A
13             t_finish ← min f_t(A)
14             for e ∈ A do
15                 if f_t(e) == t_finish then
16                     F ← F ∪ e
17                 end
18                 f_t(e) ← f_t(e) − t_finish
19             end
20             t ← t + t_finish
21             R ← A \ F
22             F ← ∅
23         else
24             N ← Neighbor(R, G')
25             G'' ← G' − N
26             A ← MaxMIS(G'')
27             SF ← SF ∪ (A, t)
28             G' ← G' − A
29             A ← A + R
30             t_finish ← min f_t(A)
31             for e ∈ A do
32                 if f_t(e) == t_finish then
33                     F ← F ∪ e
34                 end
35                 f_t(e) ← f_t(e) − t_finish
36             end
37             t ← t + t_finish
38             R ← A \ F
39             F ← ∅
40         end
41     end
42 end
```

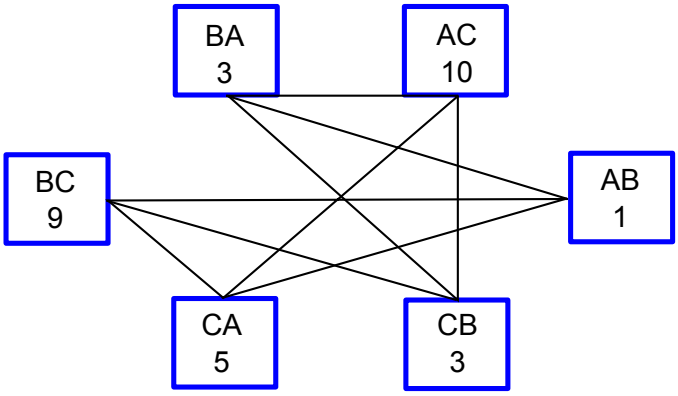


Figure 3.4: Conflict graph for the example topology shown in Figure 1.3

set $\mathcal{A}$ represents a group of *active links* that start their transmission at time t.

- *Line 1-6*: Initially, t is set to 0. The set SF is empty. The sets $\mathcal{F}$ and $\mathcal{R}$ represent the group of finishing links and remaining links respectively. Sets $\mathcal{A}$, $\mathcal{F}$ and $\mathcal{R}$ are initially empty.

- *Line 7*: Function ConflictG() computes the corresponding conflict graph G' from $G(V, E)$; see Figure 3.4.

- *Line 9-12*: When $\mathcal{R}$ is an empty set, the function MaxMIS() performs graph coloring on G' to find the MISs: $\{e_{AB}, e_{AC}\}$, $\{e_{BA}, e_{BC}\}$ and $\{e_{CA}, e_{CB}\}$. A-TxRx then chooses the MIS containing the most links. Here, as all three MISs are of the same size, A-TxRx randomly chooses $\{e_{AB}, e_{AC}\}$. Thus, e_{AB} and e_{AC} are chosen as *active links* and included in the set $\mathcal{A}$, meaning all links in this MIS start to transmit at $t = 0$. Then $\mathcal{A}$ and the corresponding t are recorded in $SF = (\{e_{AB}, e_{AC}\}, 0)$. The graph G' is updated by removing the links in $\mathcal{A}$ from G'. In this case, the scheduler removes link e_{AB} and e_{AC}, i.e., node AB, AC and its incident links in Figure 3.4.

- *Line 13-19*: Among all air-time of links in $\mathcal{A}$, A-TxRx sets the shortest one as t_{finish}. The link(s) with the shortest air-time is (are) included in the set $\mathcal{F}$. The reason is that these links, amongst those in $\mathcal{A}$, will finish the earliest. In this example, $t_{finish} = 1$, which corresponds to the air-time of link e_{AB}.

Hence, the set $\mathcal{F}$ contains e_{AB}. Also, the scheduler updates the air-time of the links in $\mathcal{A}$ by subtracting t_{finish} from the original air-time $f_t(e)$. For example, $f_t(AC) = 10 - 1 = 9$, meaning that at this point in time, link e_{AC} has transmitted for one time unit, but it still needs another nine time units to complete its transmission.

- *Line 20*: The variable t becomes $t = 0 + 1 = 1$ because one time unit has been used for links in $\mathcal{A}$ to operate.

- *Line 21-22*: The set $\mathcal{R}$ consists of links that are in $\mathcal{A}$, but not in $\mathcal{F}$. Afterwards, A-TxRx clear set $\mathcal{F}$. In this example, the set $\mathcal{R}$ contains link e_{AC} since it has not finished transmitting. At this point, since $\mathcal{R}$ is not an empty set, A-TxRx restarts from Line 24.

- *Line 24-25*: the function Neighbor() returns all interfering nodes of links in $\mathcal{R}$. A-TxRx copies G' to a new graph G'', and removes $\mathcal{N}$ from G''; referring to the example, this means e_{BA}, e_{CA} and e_{CB} are removed from G''. This step is essential because it ensures that newly activated links do not interfere with the links in $\mathcal{R}$.

- *Line 26-28*: A-TxRx then calls MaxMIS() on G'' to derive links that can be added into set $\mathcal{A}$ without any conflict, which is link e_{BC} in this case. As a result, $SF = (\{e_{AB}, e_{AC}\}, 0) \cup (\{e_{BC}\}, 1)$. The graph G' is updated by removing *active links* e_{BC} from G'.

- *Line 29*: All the links that are transmitting at this point in time are e_{AC} and e_{BC}.

- *Line 30-36*: The shortest air-time of links in $\mathcal{A}$ is $f_t(AC) = f_t(BC) = 9$. The variable t_{finish} is then set to 9. Thus, link e_{AC} and e_{BC} become *finishing links* in $\mathcal{F}$. As before, the air-time of links are updated by subtracting t_{finish} from the original air-time $f_t(e)$. In this case, $f_t(BC) = f_t(AC) = 9 - 9 = 0$, meaning that at this point in time, both links finish their transmission.

- *Line 37-39*: The variable t becomes $t = 1+9 = 10$. The set $\mathcal{R}$ becomes empty. Then clear set $\mathcal{F}$. At this stage, since there are no links in $\mathcal{R}$, A-TxRx repeats from Line 10.

- *Line 10*: A-TxRx executes MaxMIS() on G'. The MISs are $\{e_{BA}, e_{CA}\}$ and $\{e_{CA}, e_{CB}\}$. The scheduler selects $\{e_{BA}, e_{CA}\}$ into $\mathcal{A}$ arbitrarily. The process continues until t reaches time 13. At this time, G' becomes empty, meaning that all the links in the network have been scheduled. The program terminates and the output is $SF = (\{e_{AB}, e_{AC}\}, 0) \cup (\{e_{BC}\}, 1) \cup (\{e_{BA}, e_{CA}\}, 10) \cup (\{e_{CB}\}, 13)$. After another three time units, which is the air-time of the last activated link e_{CB}, all the links in the network have transmitted once. Thus, for Figure 1.3, A-TxRx generates a superframe with length of 16, reducing the superframe length generated using 2P by almost 41%. The time line of the schedule is shown in Figure 3.5.

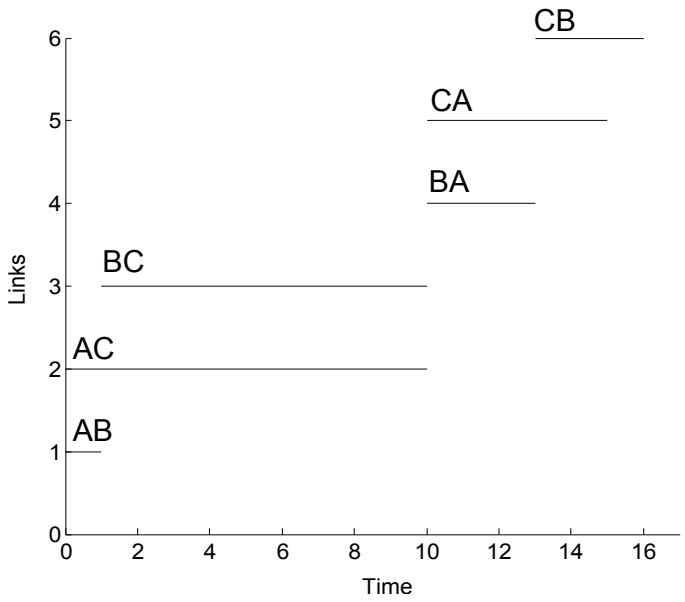


Figure 3.5: Schedule timeline for Figure 1.3

3.3.1 Opportunistic Links

Network capacity can be improved by adding "opportunistic links". These links are defined as those that can be allocated additional transmit opportunities without

interfering existing links. In the aforementioned example, note that in $Time = [15, 16]$, only link e_{CB} is transmitting. No other links are activated because all links have transmitted once. In fact, link e_{AB} or e_{CA} can also be activated as an opportunistic link as they do not interfere currently active links. Hence, A-TxRx can select e_{AB} because it has the air-time of one unit and more importantly, adding it does not change the superframe length. On the other hand, if the scheduler activates e_{CA} at $Time = 15$ with air-time of five time units, it will expand the superframe length from 16 to 20.

To add opportunistic links, the following improvement is made to A-TxRx. After constructing a new MIS consisting of new links to be added into the superframe in line 10 or line 26 of Algorithm 1, delete all the links in the new MIS and their neighbors from the original conflict graph to obtain a graph where the previously scheduled links are included. Next, color this new graph to obtain the largest MIS. This MIS contains all the links that can be activated without causing interference. Then choose only the links with an air-time that does not extend the superframe length. Hence, the derived schedule is not only a feasible one, but also has higher network capacity.

A question that arises is why only links that have been activated previously can be opportunistic links. In line 12 and 28 of Algorithm 1, as soon as a link is included in the chosen MIS and selected to transmit, it is deleted from G'. Thus, when MaxMIS() is called on graph G' in line 10 and 26, the resulting $\mathcal{A}$ already contains the most non-interfering links that have yet to be activated. None of the links in G' can be scheduled because it interferes with at least one link in $\mathcal{A}$ as $\mathcal{A}$ is an MIS. Therefore, only by searching the links that are not in G', i.e., links that have been activated at least once, can A-TxRx find some other non-interfering links and use them as opportunistic links.

3.3.2 Greedy A-TxRx

In A-TxRx, the function MaxMIS() in Line 10 and 26 first performs graph coloring on the conflict graph G' in order to find the MISs. Then the MIS with the maximum cardinality is chosen as the set of *active links* $\mathcal{A}$. However, graph coloring is a time-consuming operation; see Section 3.4. Thus, this section presents a modified A-TxRx which is denoted as A-TxRx$_{Greedy}$, where the function MaxMIS() in Line 10 and 26 is replaced with Greedy(). This function Greedy() constructs $\mathcal{A}$ by greedily searching all the links in E. To be specific, A-TxRx visits each $e \in E$ one after another. Link e is added to the set $\mathcal{A}$ if e is not conflicting with any link that is already included in $\mathcal{A}$. Otherwise, link e is not selected to join $\mathcal{A}$. In this way, the computation of this algorithm is facilitated while slightly compromised the performance in terms of superframe length and concurrent number of active links which is evaluated in Section 3.6.

3.4 Analysis

This section computes the running time complexity of A-TxRx for an arbitrary graph G with $|V|$ nodes, whereby the number of edges $|E|$ of G is upper bounded by $|V|(|V|-1)$, i.e., G is fully connected. In particular, there are the following results.

Theorem 1. *The running time complexity of A-TxRx is* $\mathcal{O}(|V|^5)$.

Proof. All lines of A-TxRx take $\mathcal{O}(1)$ except for lines 7, 8, 10, 13, 14, 24, 26, 40 and 31. In line 7, the function ConflictG() takes $\mathcal{O}(|V|^2)$ times to convert the original graph G into a conflict graph G'. The reason is that every edge in G is considered as a vertex in G' and is connected to its interfering links. Consequently, the number of vertices in G' is equal to the number of edges in G, which is bounded by $|V|(|V|-1)$. Similar for line 8, the time complexity is also $\mathcal{O}(|V|^2)$ because it considers the number of vertices in G'. In line 10 and 26, the process of graph coloring is performed. According to [96], the smallest-last graph coloring algorithm

has a time complexity of $\mathcal{O}(|V'|+|E'|)$ for $G'(V', E')$. To calculate the total number of edges in G', let us assume the worst case, whereby G is fully connected. For any link, say e_{AB}, there are $2|V| - 3$ number of links originating from node B and directing to node A that are in conflict with e_{AB}. Hence, the total number of edges in G' is calculated as $(2|V| - 3) \times |V|(|V| - 1)$. As a result, the time complexity for line 10 and 26 is $\mathcal{O}(|V|^2 + |V|^3) = \mathcal{O}(|V|^3)$. For line 13, 30, 14 and 31, in the worst case, every vertex $v' \in V'$ in G' is linked with only one vertex which represents its corresponding opposite direction link, and thus the size of the largest MIS, used as $\mathcal{A}$, is equal to $\frac{1}{2}|V|(|V| - 1)$. Similarly, in the worst case, the size of $\mathcal{R}$ is equal to $\frac{1}{2}|V|(|V - 1|)$. Therefore, line 12, 30, 14, 31 and 24 take at most $\mathcal{O}(|V|^2)$ time. Based on the above calculation, A-TxRx has a time complexity of $\mathcal{O}\Big(|V|^2(|V|^3)\Big) = \mathcal{O}(|V|^5)$. □

Theorem 2. *A-TxRx produces a collision-free schedule.*

Proof. By using prove by contradiction, suppose there are two links transmitting *and* receiving at the same time; specifically, e_{AB} and e_{xA} or e_{AB} and e_{By} are transmitting concurrently, where x can be any node but A, y can be any node but B. So there are two cases:

Case 1: These two interfering links start to transmit at the same point in time. In this case, the two links are activated in line 10. After the graph coloring process, only links in one independent set are activated. Recall that, in an independent set of a graph, no two vertices of the graph are adjacent. This contradicts that the two links are interfering with each other. This leaves the second case.

Case 2: One of these two conflicting links, say e_{xA} (or e_{By}) starts to transmit while the other link, say e_{AB} has not finished its transmission. It indicates that the newly added link is activated in line 26, which applied graph coloring to a new graph G''. Note that in line 25, all the remaining links including e_{AB} and their neighboring links are deleted from G''. Thus, any link that conflicts with link e_{AB} does not exist in graph G''. Hence, it contradicts that e_{xA} (or e_{By}) are derived from line 25. As a

result, it is impossible for A-TxRx to derive a schedule with interference. □

Theorem 3. *A-TxRx generates a link schedule that is no longer than* $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$ *for a bipartite graph topology* $G(V, E)$.

Proof. Let V_1 and V_2 be the two subsets of a bipartite graph $G(V, E)$ such that nodes in V_1 (V_2) have links only to nodes in V_2 (V_1). Line 7 of A-TxRx generates the conflict graph $G'(V', E')$ from G. Notice that one can generate two MISs, i.e., $\mathbb{A}$ and $\mathbb{B}$, from G'; $\mathbb{A}$ ($\mathbb{B}$) contains links emanating from nodes in V_1 (V_2). Let i_1 (i_2) be a link emanating from V_1 (V_2) with the longest air-time, i.e., $f_t(i_1) = l_t^{\mathbb{A}}$; similarly, $f_t(i_2) = l_t^{\mathbb{B}}$. Note that there can be multiple number of links i_1 and i_2. Consider three possible cases: (i) all links in $\mathbb{A}$ ($\mathbb{B}$) have the same air-time, i.e., each link in $\mathbb{A}$ ($\mathbb{B}$) has an air-time $l_t^{\mathbb{A}}$ ($l_t^{\mathbb{B}}$), (ii) links i_1 and i_2 are interfering with each other, and (iii) links i_1 and i_2 are not interfering with each other.

For case (i), A-TxRx generates a schedule in which links in $\mathbb{B}$ are activated after all links in $\mathbb{A}$ have completed their air-times, resulting in $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$. Specifically, Line 10-12 generate MIS $\mathcal{A} = \mathbb{A}$ and schedule all links in $\mathbb{A}$ starting at time $t = 0$ for $l_t^{\mathbb{A}}$ time unit,and $G' = \mathbb{B}$. Line 14-19 generate $\mathcal{F} = \mathbb{A}$ because all links have the same $t_{finish} = l_t^{\mathbb{A}}$, and thus Line 20-21 obtain $t = l_t^{\mathbb{A}}$, and $\mathcal{R} = \emptyset$. Therefore, in the second iteration of the loop in Line 8, Line 10-12 generate MIS $\mathcal{A} = \mathbb{B}$ and schedule all links in $\mathbb{B}$ starting at time $t = l_t^{\mathbb{A}}$ for $l_t^{\mathbb{B}}$ time unit, and $G' = \emptyset$. Similar to for $\mathbb{A}$, this iteration produces $\mathcal{F} = \mathbb{B}$, and $\mathcal{R} = \emptyset$. Thus the algorithm completes with a schedule that starts at $t = 0$ for all links in $\mathbb{A}$ for $l_t^{\mathbb{A}}$ time unit, and completes with all links in $\mathbb{B}$ that start from $t = l_t^{\mathbb{A}}$ for $l_t^{\mathbb{B}}$ time unit. Since the schedule completes at time $l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$, for this case, A-TxRx generates the schedule with a superframe length of $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$.

For case (ii), some links other than i_2 can be activated at the same time with links in $\mathbb{A}$ as long as they do not interfere. However, since i_2 and i_1 are interfering links, i_2 can be activated only after i_1 has completed its $l_t^{\mathbb{A}}$ air-time; i_2 completes its air-time at time $l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$, meaning the superframe length for this case is $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$

because all links in $\mathbb{A}$ ($\mathbb{B}$) complete no later than time $l_t^{\mathbb{A}}$ ($l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$). Specifically, Line 10-12 generate MIS $\mathcal{A} = \mathbb{A}$ and schedule all links in $\mathbb{A}$ at time 0. Since $\mathcal{R}$ contains at least i_1, and $G' \neq \emptyset$, in the second iteration, Line 24-27 schedule links in $\mathbb{B}$ that do not interfere with the remaining links in $\mathcal{R}$ of $\mathbb{A}$ that have air-time larger than t_{finish}. For Line 30, consider two possible cases: (ii.1) at least one additional link from $\mathbb{B}$ has an air-time longer than the remaining air-time of i_1; (ii.2) the remaining air-time of i_1 is the longest among links in the new MIS. For case (ii.1), A-TxRx iterates the else in Line 23, and eventually will generate an MIS that contains only links in $\mathbb{B}$, including i_2, at time $l_t^{\mathbb{A}}$. At this stage, Line 24-26 generate $\mathcal{N} = \emptyset$ since nodes in $\mathcal{R} = G' \subseteq V_2$ and the nodes are not neighbors of the others, $G'' = \emptyset$, and $\mathcal{A} = \emptyset$; thus Line 26 does not add more links to SF, links that start at time $l_t^{\mathbb{A}}$ will complete for at most $l_t^{\mathbb{B}}$ time unit, meaning the superframe length is $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$. For case (ii.2), A-TxRx will continue iterating from Line 24 until reaching case (ii.1) or the remaining air-time of i_1 is zero. In either case, at time $l_t^{\mathbb{A}}$, $\mathcal{R}$ will eventually contain links only from $\mathbb{B}$, and as described before, the links will be activated no longer than $l_t^{\mathbb{B}}$ time unit, giving the superframe length of $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$.

For case (iii), link i_2 can be activated at the same time with link i_1 as long as its interfering link has completed its air-time. Thus, for this case, A-TxRx generates schedule with superframe length $\delta(G) < l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$. Specifically, when A-TxRx reaches Line 24, $\mathcal{N}$ does not contain i_2, and thus the SF in Line 27 contains both i_1 and i_2, which eventually reduces the superframe length of the schedule to less than $l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$ without interfering links.

Based on the above analysis, when i_2 and i_1 are interfering links, i_2 can be activated only after i_1 has completed its $l_t^{\mathbb{A}}$ air-time; i_2 completes its air-time at time $l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$, meaning the superframe length for this case is $\delta(G) = l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$ because all links in $\mathbb{A}$ ($\mathbb{B}$) complete no later than at time $l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$. In general, an arbitrary bipartite graph falls into case (iii), and thus A-TxRx should generate schedule with superframe length $\delta(G) < l_t^{\mathbb{A}} + l_t^{\mathbb{B}}$. □

3.5 Research Methodology

In this section, MatGraph [97] is used to evaluate the performance of A-TxRx. It is a Matlab toolkit to work with simple graphs. Specifically, the following two functions are used: (i) *color()*, which performs a vertex coloring on the graph, and (ii) *random_regular()*, which takes as parameter the degree of each node. In all experiments, nodes are stationary and randomly located on a square area. Note that channel error is not taken into account. In practice, retransmissions due to channel errors can be accounted for by dimensioning the transmission time accordingly. Moreover, directional transmissions tend to have high gains. Otherwise, links with poor channel condition can be omitted from the topology.

This section studies the impact of the following parameters on the performance of A-TxRx: node density, transmission radius, node degree and selected MISs, as well as its running time. The number of nodes ranges from 5 to 40 with an interval of 5. The transmission radius ranges from 10m to 130m with an interval of 20m. The network area ranges from $50m \times 50m$ to $250m \times 250m$. The degree of each node varies from 2 to 10, assuming a total of 11 nodes. Five experiments are conducted with one change to the network configuration while others are fixed; this will be made specific in the result sections later. The results are an average of 20 simulation runs, each with a different topology.

A-TxRx, which represents both A-TxRx$_{GC}$ and A-TxRx$_{Greedy}$, is compared against 2P-1slot, 2P-node and JazzyMAC [26], where A-TxRx$_{GC}$ is the algorithm adopts graph coloring, and A-TxRx$_{Greedy}$ uses greedy searching. Specifically, for both 2P-1slot and 2P-node, the scheduling is done on time slot bases. The slot size is set to the longest air-time among the active links in that slot. For 2P-1slot, color the conflict graph of a topology to yield its MIS. Then, activate all the links in the MIS to transmit in the first time slot. These links then start to receive in the following time slot. For example, in slot i, where i is an odd number, if e_{AB} is activated, then link e_{BA} will transmit in slot $i+1$. On the other hand, 2P-node performs a graph

coloring on the network graph instead of the conflict graph. After the MIS with the most nodes is determined, all nodes in the MIS transmit and become receiver in the next time slot. Then, remove the chosen nodes from the network graph and repeat the above process until all links are activated at least once. JazzyMAC also exploits TDMA but its slot length is dynamic. It is initialized centrally and then works according to the following fundamental rules: each node holds one token for each of its links. When a node finishes its transmission on a link, it passes the corresponding token to the other end of the link. A node becomes a transmitter when it holds the token for all its incident link(s). Other details of JazzyMAC can be found in [26].

In each experiment, the following metrics are collected with error bars indicate 95% confidence interval:

- *Superframe length.* This is the total time duration for each link to transmit at least once.

- *Number of concurrent active links.* It corresponds to the average number of links that are operating concurrently at each point in time. This also includes the number of opportunistic links enabled outlined in Subsection 3.3.1. This metric is significant because it reflects the capacity of the WMN.

- *Computation time.* This is the time required for each algorithm to calculate the schedule for a given topology on a computer with an Intel Core i7 with 6 GB RAM.

3.6 Results

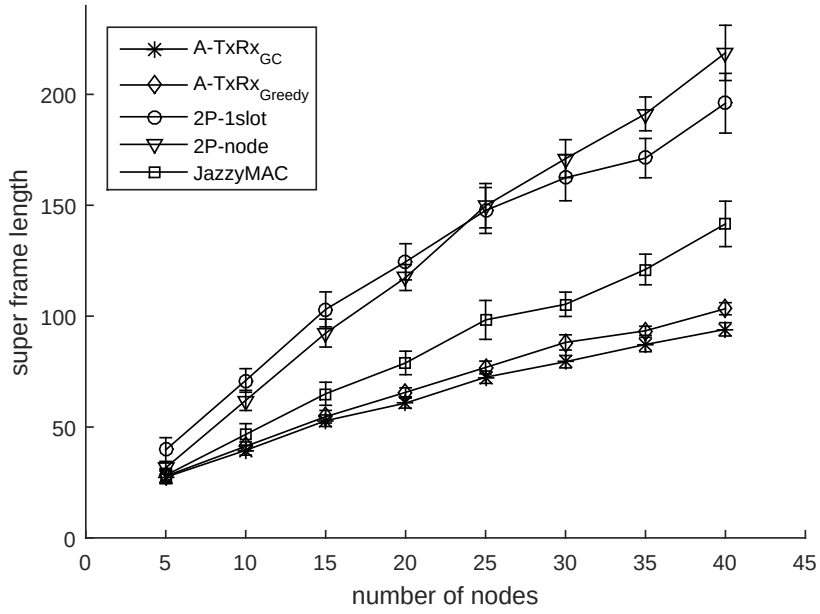
The following sections present the results from experiments concerning node density, transmission radius, node degree, running time and impact of the selected MIS.

3.6.1 Node Density

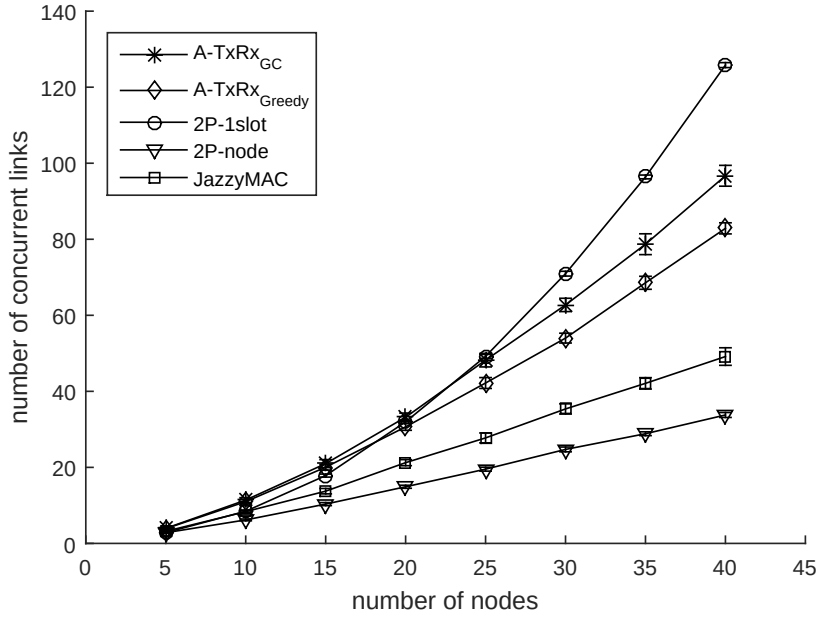
The first experiment studies the effect of node density on superframe length and number of concurrent links. The number of nodes ranges from five to 40. The transmission range of each node is set to 70.

From Figure 3.6(a), we see that the superframe length increases when more nodes are added. The reason is that as the number of nodes increases from five to 40, the number of links grows from 15 to 1159. Thus, there are more interfering links at each point in time (or during every time slot). Additionally, in Figure 3.6(a), we can also see that the superframe length of A-TxRx is significantly shorter than 2P-1slot and 2P-node, i.e., less than half as long. The reason is that, in 2P-1slot and 2P-node, as long as the MIS is determined, all links in the MIS are regarded as a group of links that are assigned with the transmission right in the following time slot for the same period of time to operate and finish their transmission. However, among this group of links, some links may end their transmission sooner, i.e., have a shorter air-time than the allocated slot duration. As a result, part of the channel will be idle. For A-TxRx, as long as one link finishes transmission, the largest set of new non-conflicting links are activated immediately to make maximum use of transmitting channel.

In Figure 3.6(b), except for JazzyMAC, 2P-1slot, 2P-node and A-TxRx are extended by implementing opportunistic scheduling. Note that opportunistic links do not affect the superframe length. As shown, 2P-1slot produces about 25% more concurrent links as compared to A-TxRx, especially for high density networks, i.e., when the number of nodes reaches 40. The key reason is because 2P-1slot can add more opportunistic links due to the longer slot time. Both A-TxRx and 2P-1slot significantly outperform 2P-node and JazzyMAC. The reason 2P-node generates much less concurrent links is that 2P-node select transmitting links in a node bases. If one incident link of a node is determined to be an interfering link, then none of the other incident links of this node can be activated to transmit even if such links



(a)



(b)

Figure 3.6: Superframe length (a) and average number of active links at each time point (b) under different node densities

are non-interfering. For JazzyMAC, it also has less concurrent links because it is impossible to extend JazzyMAC with opportunistic scheduling.

To quantify the benefits of opportunistic scheduling, the experiments for A-TxRx are repeated. Figure 3.7 compares the number of concurrent links with and without opportunistic scheduling for A-TxRx. We can see that when opportunistic scheduling is implemented, there are about 20% more links as an increasing number of nodes are added whilst keeping the network area fixed.

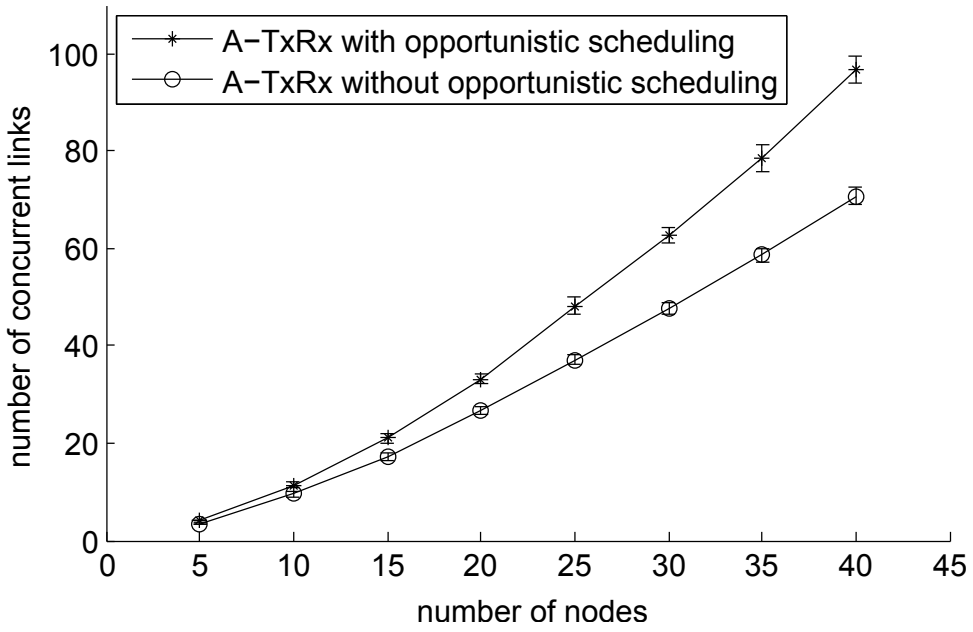


Figure 3.7: Improvement in the number of concurrent links with opportunistic scheduling

Furthermore, by comparing A-TxRx$_{GC}$ and A-TxRx$_{Greedy}$, we can see that for a small amount of nodes, i.e., nodes less than 15, these two algorithms have exactly the same performance. However, as the number of nodes increases from 15 to 40, A-TxRx$_{GC}$ shows a slightly superior performance than A-TxRx$_{Greedy}$ of around 8% shorter superframe length and around 10% more concurrent links. This is because A-TxRx$_{GC}$ tends to activate the largest MIS among all MISs generated by graph coloring. While in A-TxRx$_{Greedy}$, the set of activated links is simply a random MIS. Hence, the $\mathcal{A}$ of A-TxRx$_{GC}$ is most likely to contain more links than that of A-TxRx$_{Greedy}$. For this reason, A-TxRx$_{GC}$ activates more links in each iteration, and

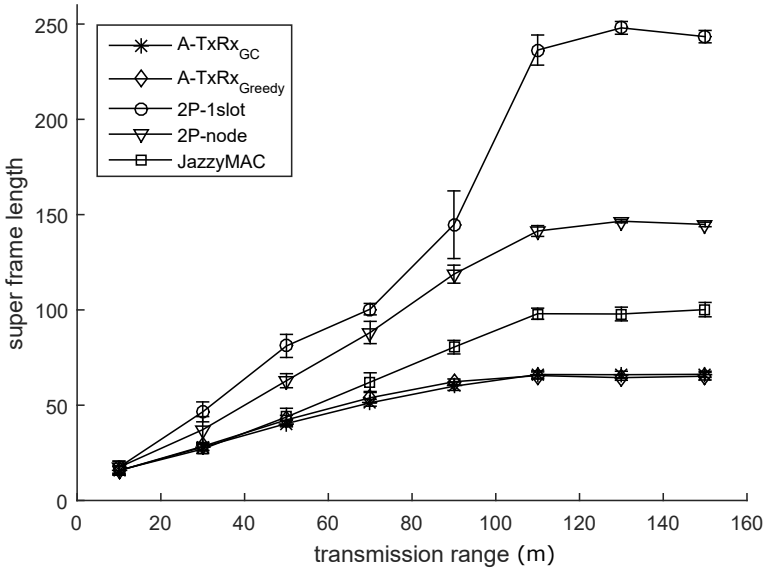
thus leads to a shorter superframe length.

3.6.2 Transmission Radius

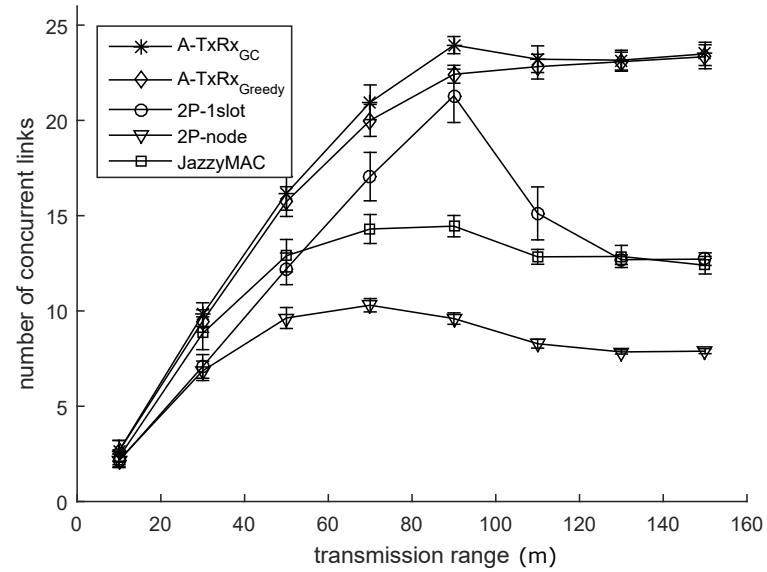
This experiment evaluates the performance of all algorithms when the transmission range of nodes is changed. There are 15 nodes located on a square area of $100 \times 100m^2$. The air-time range is from one to 10 time units.

Figure 3.8(a) shows that when the transmission range increases from 10m to 90m, the superframe length of all four algorithms increases linearly. The superframe length of A-TxRx increases by 12 units after the transmission range is increased by a step size of 40m. Correspondingly, 2P-1slot, 2P-node and JazzyMAC experienced an approximate increase of 30, 20, and 16 time units. For small transmission range, i.e., 10m, the possibility of establishing a link between two nodes is small, and thus there are only an average of 6.8 links in total. On the other hand, increasing the transmission range to 110m increases the possibility of establishing links, which also increases the superframe length since more links need to be scheduled. For the range, there are on average 208.7 links in the network, and thus the network becomes almost fully connected, whereby every node is connected to all other nodes. Thus, transmission ranges from 110m to 150m produce almost the same conflict graph. As a result, the superframe length of A-TxRx, 2P-1slot, 2P-node and JazzyMAC becomes almost constant at values of 66, 243, 145 and 99 respectively. We can see that the difference between the superframe length of A-TxRx and that of the other three algorithms more than doubles at transmission range of 50m and 110m. This shows that A-TxRx is particularly advantageous over the other three algorithms when the transmission range is large.

Figure 3.8(b) shows the average number of links that are active at each point in time under different transmission range. We can see that the number of concurrent links of A-TxRx increases and reaches its peak value of 24 when the transmission range is 90m. With increased transmission radius, the graph becomes fully con-



(a)



(b)

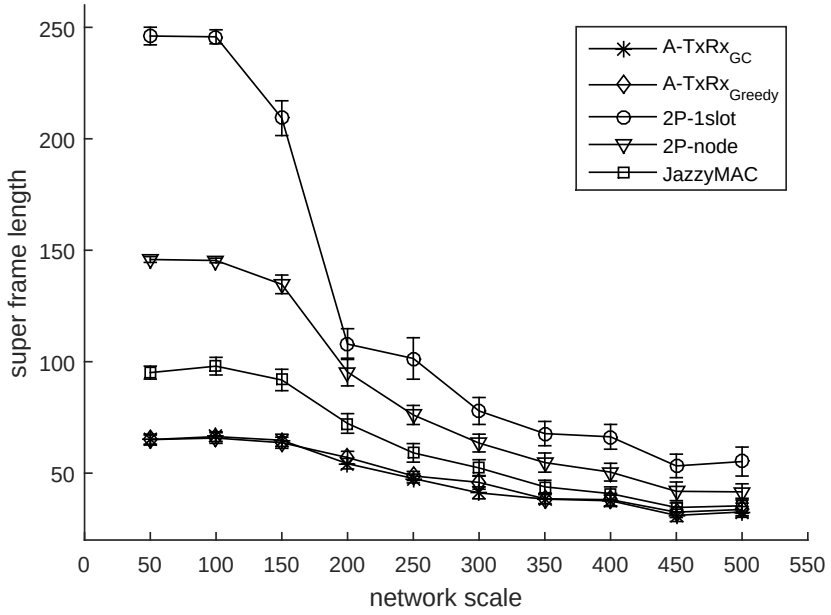
Figure 3.8: Superframe length (a) and average number of active links at each time point (b) under different transmission radii

nected, resulting in the number of concurrent links fixes at around 23, regardless of the randomness in nodes distribution. However, for 2P-1slot, after it reaches a peak value of 21 at 90m, the number of links notably decreases to 15, and then becomes steady around 12. This is because for a fully connected graph, excluding opportunistic links, when 2P-1slot first performs graph coloring, only 14 links can be activated. To be specific, these 14 links originated from 14 different nodes and are directed to the same destination node, say node A. After removing these 14 links, node A becomes disconnected from the network graph. As a result, during the second graph coloring process, there will only be 13 active links. Hence, the number of transmitting links decreases from 14 links (slot one) to one link (last slot). Thus, the average number of concurrent links is less than $\frac{\sum_{i=1}^{i=14} i}{14} = 7.5$. Although opportunistic links can be added to increase the total number of links in each slot, only the links with air-time that is less than the slot duration can be selected and activated in the current slot.

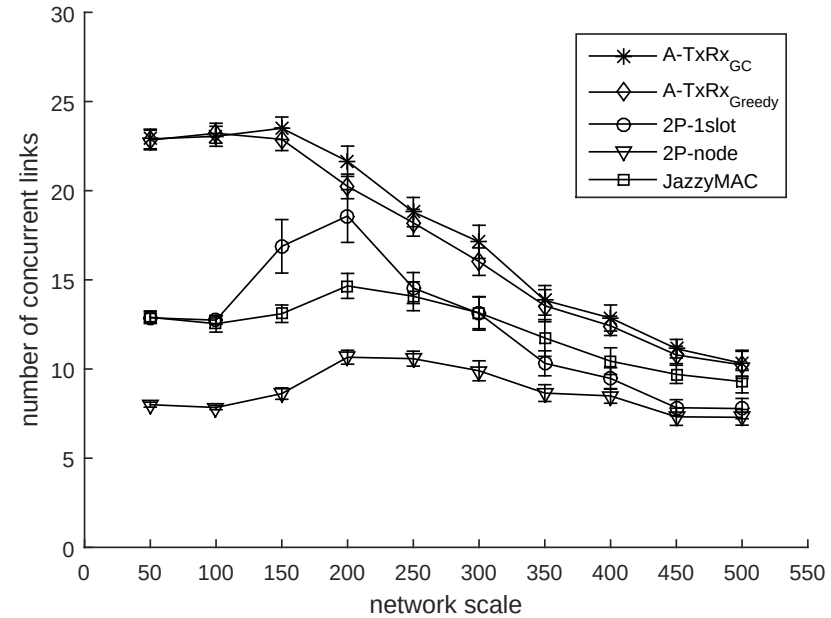
3.6.3 Network Scale

This experiment studies the performance of each algorithm when the network coverage varies from $50 \times 50m^2$ to $500 \times 500m^2$. The number of nodes is set to 15 with a transmission range of 150m. The air-time range is [1, 10].

Figure 3.9(a) shows the superframe length with increasing network coverage. As the network increases from 50 to 100 square meters, the superframe length is 65, 245, 145 and 96 time units for A-TxRx, 2P-1slot, 2P-node and JazzyMAC respectively. This is because the network is fully connected under the above configuration. As the network scale increases while the transmission range being fixed, there are fewer number of links established in this network. As a result, with fewer links to schedule, the superframe length of each method becomes shorter. When the coverage increases from 100 to 500 square meters, the difference in superframe length for A-TxRx and other algorithms decreases from 73% to 22%. This means that the performance of



(a)



(b)

Figure 3.9: Superframe length (a) and average number of active links at each time point (b) under different network scales

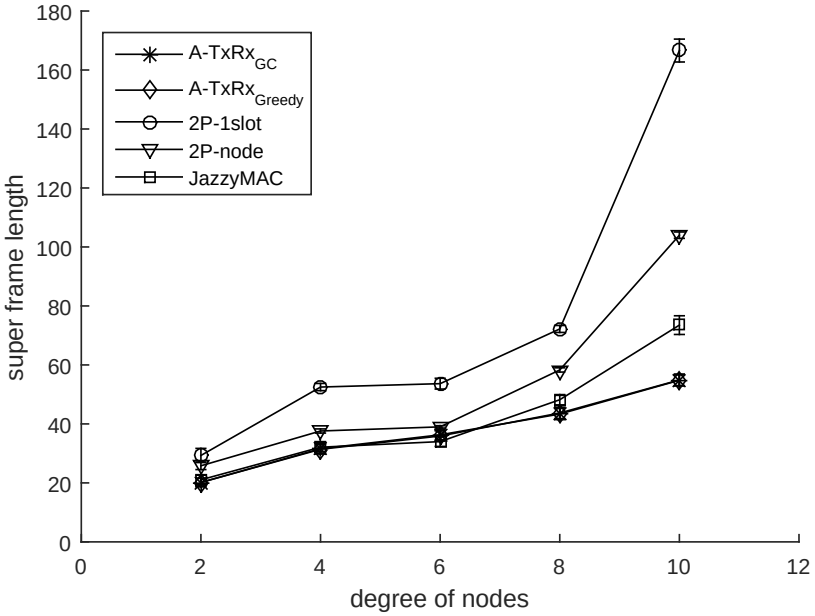
A-TxRx is superior to other algorithms, especially in smaller networks, where the distribution of nodes is denser.

Figure 3.9(b) shows the number of concurrent links for each algorithm. We can see that the number of concurrent links of 2P-1slot, 2P-node and JazzyMAC increases and reaches a peak value of 18, 10 and 15, respectively when the network scale increases to $200 \times 200m^2$. Interestingly, A-TxRx is not affected when the graph becomes fully connected. To be specific, the number of concurrent links when using A-TxRx remains close to constant at around 23, when the square length increases from 50m to 150m.

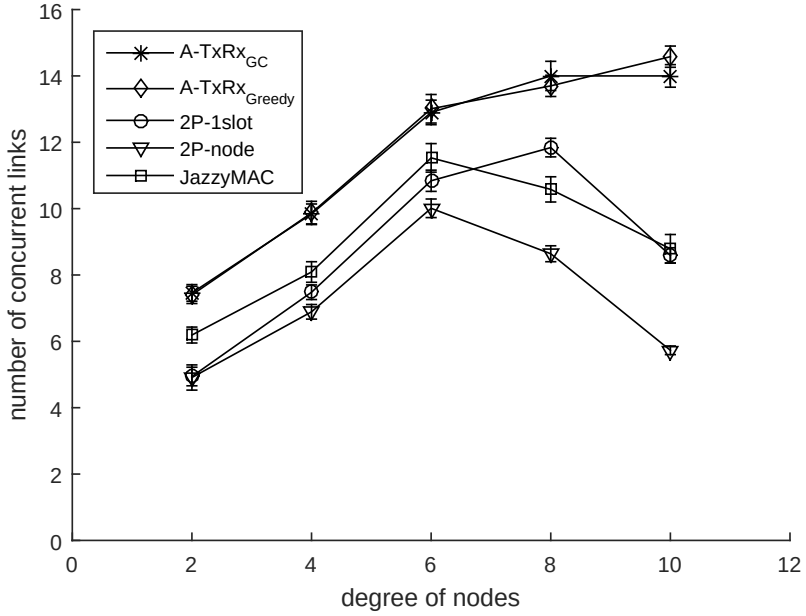
3.6.4 Node Degree

In this experiment, the number of nodes is set to 11. The degree of all nodes is the same and varies from two to 10. The aim is to study the relationship between node degree and superframe length, and also the influence of superframe length and total number of links on delays.

Figure 3.10(a) shows that the superframe length of A-TxRx increases linearly from 18 to 56 as the node degree rises. This is because the total number of links $d \times n$ is a linear function, where d represents the degree of nodes that increases from two to 10, and n is the number of nodes that is set to 11. Interestingly, we can see that when the number of degrees of each node ranges from two to eight, the superframe lengths of JazzyMAC and A-TxRx are very close to each other. Then as the node degree increases from eight to 10, the superframe length of 2P-1slot, 2P-node and JazzyMAC rises by a dramatic 130%, 80% and 46% respectively. This can be explained using Figure 3.10(b). We see that there are significantly fewer number of concurrent links for 2P-1slot, 2P-node and JazzyMAC when node degree increases from eight to 10.



(a)



(b)

Figure 3.10: Superframe length (a) and average number of active links (b) under different node degrees

3.6.5 Air-Time Range

This experiment changes the range of the assigned air-time for each link. To be specific, the value of the air-time becomes an integer within different intervals: [1,5], [1,10], ...,[1, 50]. For this experiment, 15 nodes with transmission range of 70m are placed on a $100 \times 100m^2$ area. The main goal is to study the efficiency of the algorithm when sending packets of different lengths, or packets of the same size but over different transmission data rates.

Figure 3.11(a) shows the superframe length under different air-time ranges. We can see that the superframe length of all four algorithms increases linearly with small fluctuations as the air-time assigned to each link rises. Note, as the air-time range becomes greater, the difference between the superframe length of 2P-1slot and that of A-TxRx increase from 22 to 250 time units. This means that A-TxRx is particularly beneficial when the air-time range is large. The reason is that, as the air-time range increases, the slot duration of 2P-1slot and 2P-node become greater. For example, when air-time range is $[1, 5]$, a link l with air-time of one time unit can occupy a channel without transmitting data for at most four time units. But when the range equals to [1, 50], l will employ the channel for at most 49 time units, which causes a much longer superframe length for every link to be activated once.

Figure 3.11(b) shows the average number of active links at each point in time. In general, the number of concurrent links stays the same at 21, 20, 18, 10 and 13 for A-TxRx$_{GC}$, A-TxRx$_{Greedy}$, 2P-1slot, 2P-node and JazzyMAC respectively. This is because the increase in air-time does not affect network throughput. The small variations are caused by opportunistic links added to the schedule.

3.6.6 Computation Time

This section measures and compares the computation time required for each algorithm to compute the schedule for a given topology. Figure 3.12 (a) and (b) have the same network configuration as used in subsection 3.6.1 and 3.6.4 respectively.

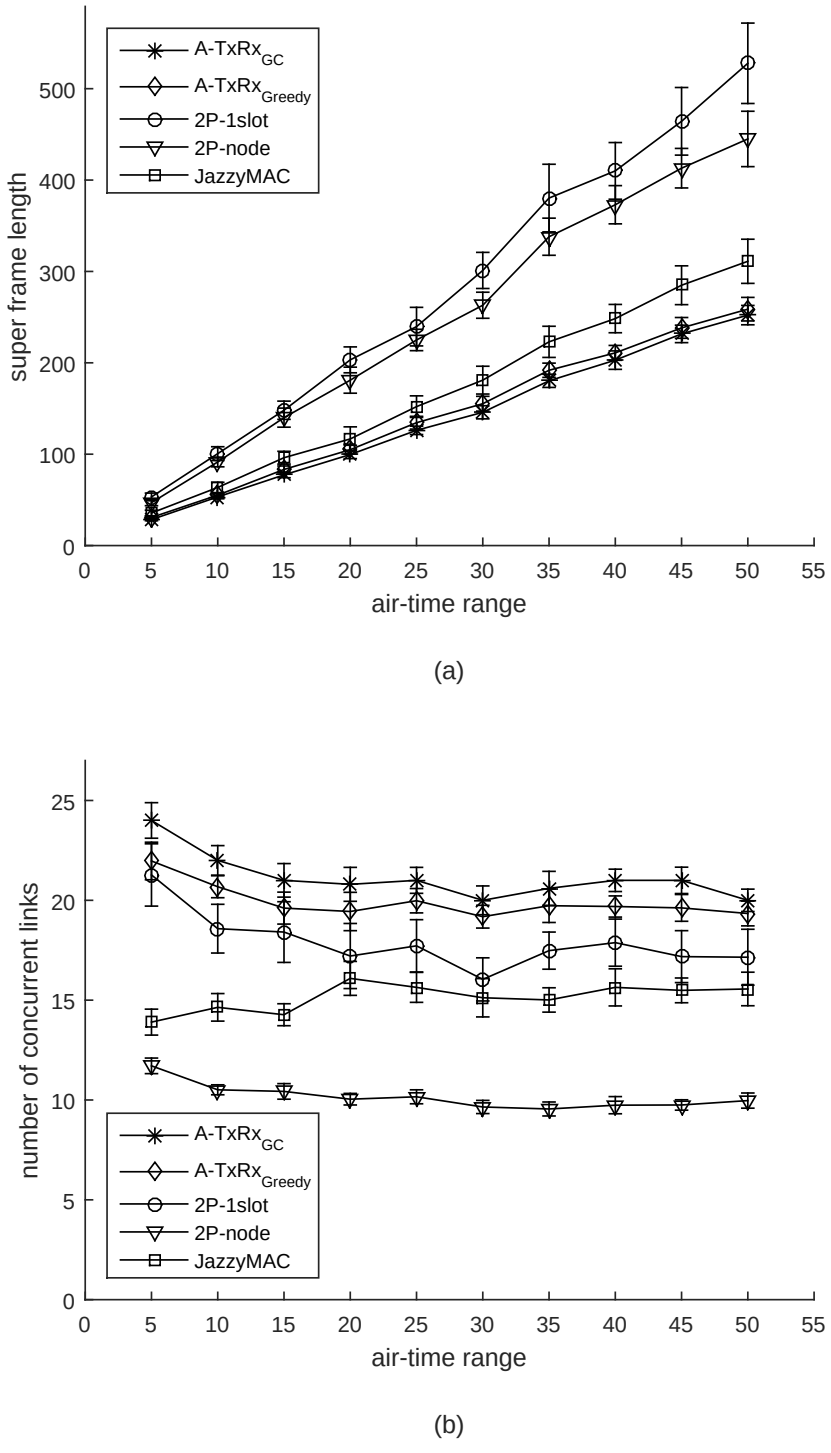


Figure 3.11: Superframe length (a) and average number of active links at each time point (b) under different air-time range

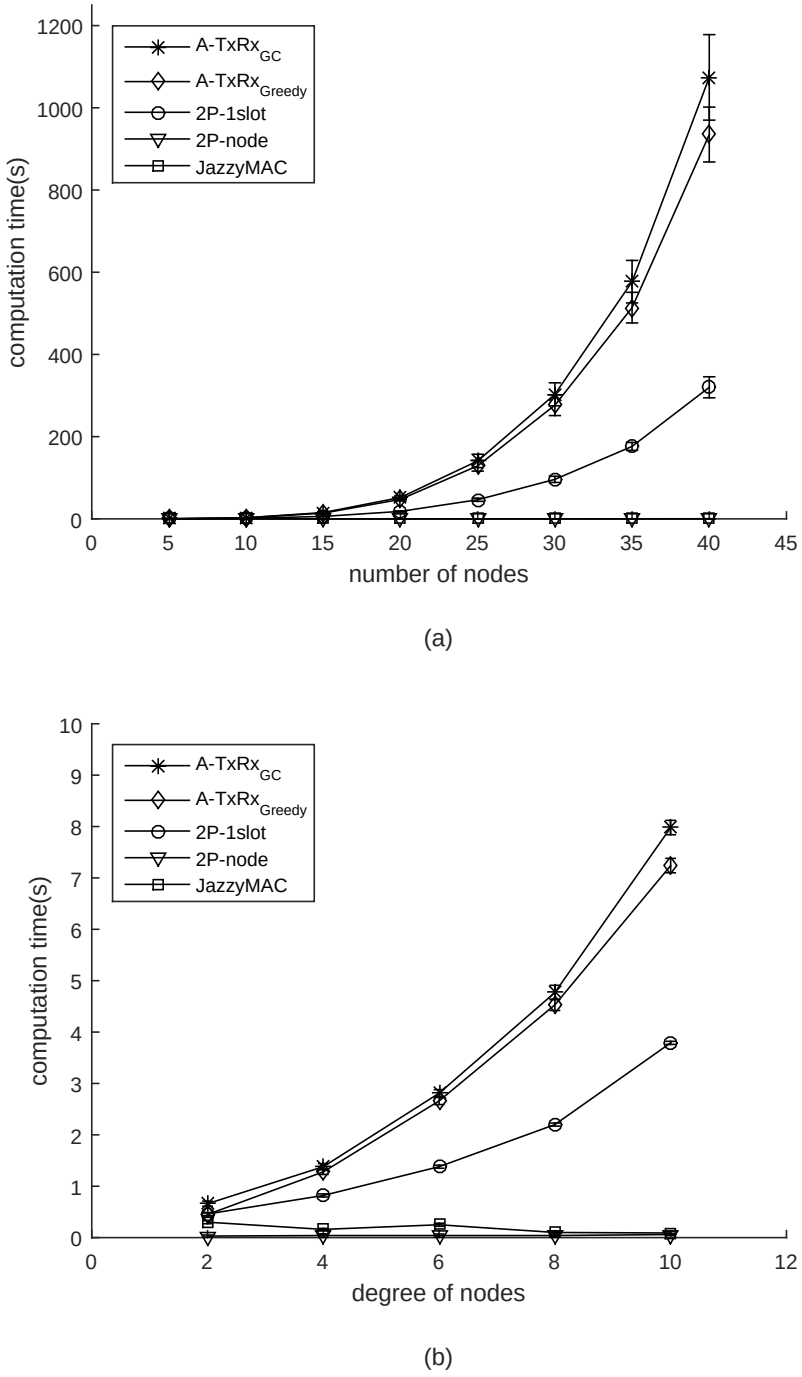


Figure 3.12: Computation time under different node densities (a) and different node degrees (b)

Figure 3.12 (a) shows the computation time of each algorithm under different number of nodes. Note, the time required for 2P-node and JazzyMAC is less than 0.1s, which is much smaller than A-TxRx. In particular, when there are 40 nodes, the computation time of 2P-node and JazzyMAC is only 0.02% of that of A-TxRx. The main reason is because 2P-node and JazzyMAC runs a graph coloring algorithm on the network topology as opposed to its corresponding conflict graph, which contains up to $|E|$ nodes and $\frac{|E|(|E|+1)}{2}$ edges. In addition, 2P-1slot also has a faster running time than A-TxRx, i.e., up to three times faster. This is because 2P-1slot only runs every other slot. Specifically, after computing the transmitters of slot i, these transmitters become receivers in slot $i+1$.

Figure 3.12(b) shows the algorithm running time with increasing node degrees. We see that the running time for all algorithms increases as the node degree rises. This is because the existence of more links. To be specific, 22 more links are established as node degree increases by two. Interestingly, the computation time for A-TxRx increases from 1s to 3s. This is due to the number of edges in the conflict graph, which increases by 30 times from 31 to 995. Hence, the graph coloring process requires more time, which incurs a longer computation time.

We can see that A-TxRx$_{Greedy}$ takes on average 10% faster than A-TxRx$_{GC}$. The reason is that the time complexity of the function Greedy() in line 10 and 26 of A-TxRx$_{Greedy}$ is $\mathcal{O}(|V|^2)$ since it visits every vertex of the corresponding conflict graph $G'(V', E')$, where $|V'| = |V|(|V|-1)$ in the worst case. However, the run time complexity of A-TxRx$_{Greedy}$ is $\mathcal{O}(|V|^4)$, which explains its faster running time.

3.6.7 Impact of choosing different MIS for A-TxRx$_{GC}$

An interesting question that arises is whether the selection of different MISs of A-TxRx$_{GC}$ have any influence on the superframe length and the number of concurrent links. To answer this question, compare the superframe length and number of concurrent links for the following cases: (i) largest MIS, which has the largest

cardinality; (ii) longest MIS, which contains the link with the longest air-time; (iii) shortest MIS; and (iv) random MIS.

Figure 3.13 (a) and (b) show the superframe length and number of concurrent links when the degree of each node increases from 2 to 10 for a network of 11 nodes. Observe that the choice of MIS does not have any significant impact on the performance of A-TxRx$_{GC}$. The reason is that A-TxRx$_{GC}$ repeatedly performs graph coloring on the updated network graph. In this way, links are almost evenly assigned into different MISs. Thus, the disparity between the size or air-time length of the MISs is too slight to make a difference. Therefore, A-TxRx$_{GC}$ is insensitive to MIS selection.

3.6.8 A-TxRx performance on bipartite graphs

The last experiment uses A-TxRx to generate schedules for linear, ring and grid topology shown in Figure 3.1, 3.2 and 3.3 respectively to support Theorem 3. Figure 3.14 plots the time line of each resulting schedule.

In case (i) of Theorem 3, equal air-time for each link is assumed. Thus, when an air-time of 5 is set to all links in the grid topology shown in Figure 3.3, the superframe length derived by A-TxRx is 10, as shown in Figure 3.14(a), which equals the result calculated by Theorem 3. For case (ii), assumed the links with longest air-times in conflict. Let us take Figure 3.2 as an example, where link e_{12} with air-time of 9 units and e_{61} with air-time of 7 units interfere with each other. The resulting superframe of the simulation in Figure 3.14(b) has a length of 16 and it equals to the theoretical bound of $9 + 7$. Finally, there is an example of a linear topology shown in Figure 3.1 to meet case (iii) of Theorem 3 where links with the longest air-times do not interfere with each other. Here, because links e_{43} and e_{21} with longest air-time 4 and 3 units respectively can transmit at the same time, the superframe length using A-TxRx is 6 (shown in Figure 3.14(c)), which is smaller than the bound of $4 + 3$.

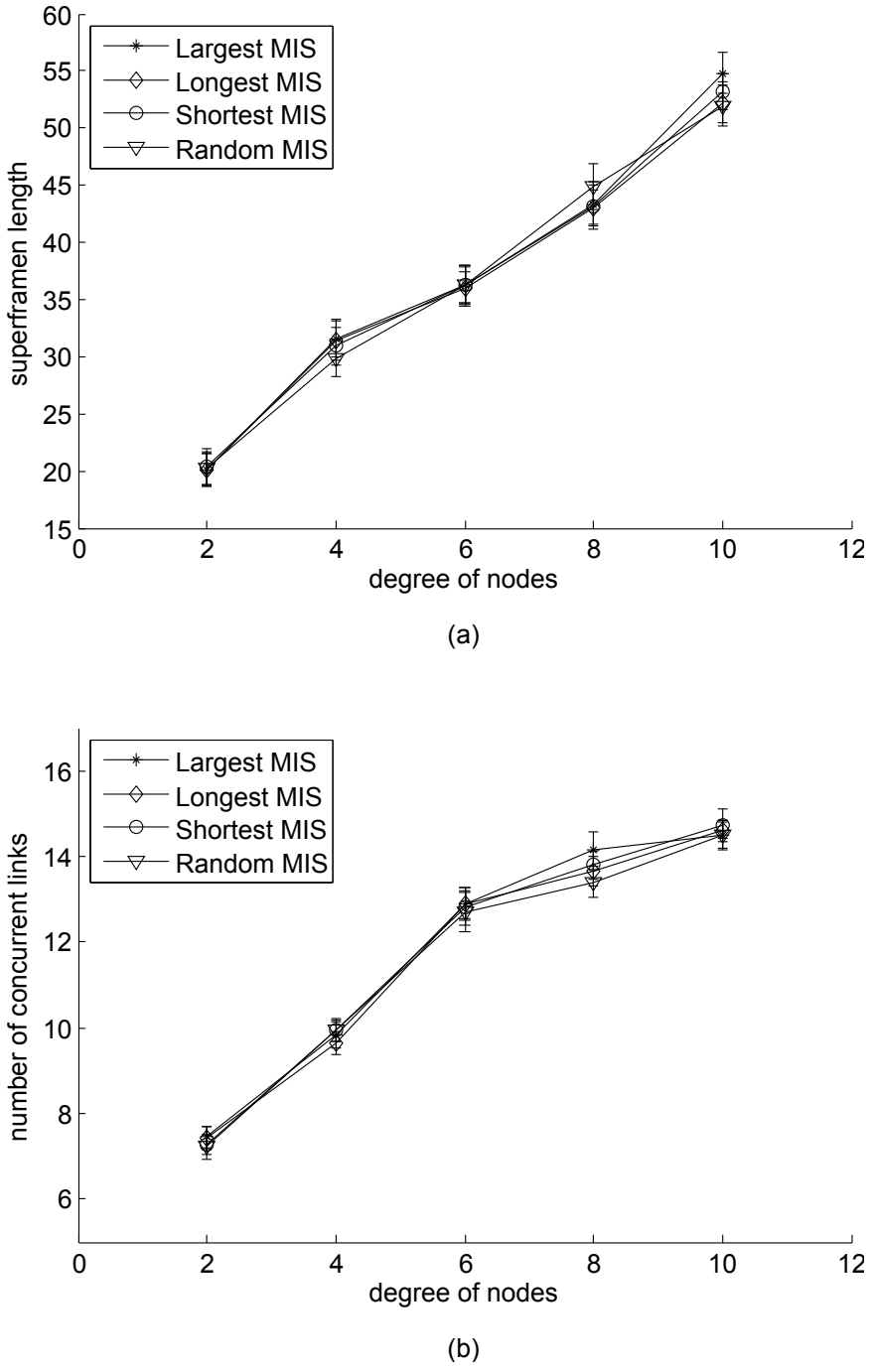


Figure 3.13: Superframe length (a) and number of concurrent links (b) under different node degrees

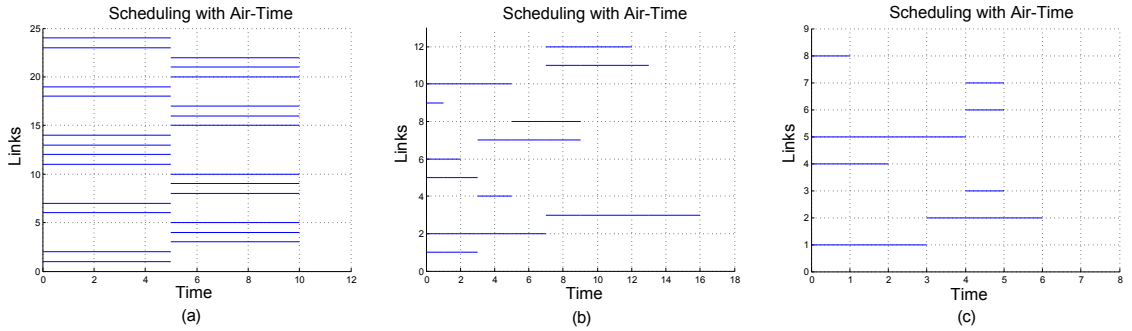


Figure 3.14: Superframe of the schedule using A-TxRx for (a) linear topology, (b) even ring topology and (c) grid topology

3.7 Conclusion

This chapter has proposed and studied a novel scheduling algorithm called A-TxRx. Its aim is to maximize the number of concurrent transmissions at any point in time; i.e., boost network capacity as well as minimize superframe length. A-TxRx is the first centralized algorithm that schedules links with different link weights on a general network topology. It activates links whenever a link finishes transmission. Links are activated if they are not in conflict with currently active links. The results show that A-TxRx yields smaller superframe lengths, and hence higher network capacity, as compared to state-of-the-art approaches. Specifically, the results show A-TxRx to have superior performance with up to 70% shorter superframe lengths and 60% more concurrent links as compared to 2P and JazzyMAC.

A key limitation of the proposed algorithm is that A-TxRx is centralized. It means that a single central node is used as a controller to monitor the air-time of every link. Based on this information, the controller activates each link in accordance to its requested air-time. However in practice, distributed algorithms outperform centralized ones in terms of scalability, lower connection setup delays, and smaller management overheads. Henceforth, the next chapter outlines a distributed algorithm whereby the scheduling task is carried out by nodes locally.

Chapter 4

A Distributed Pseudo TDMA Protocol

As shown in Chapter 2, there are a number of approaches that aim to derive the shortest feasible link schedule for a MTR WMN. These link schedulers, however, are centralized; a central controller or gateway node is required to gather topological information, construct a link schedule and disseminate it to all nodes. Consequently, these link schedulers will incur many rounds of signaling overheads, as well as large propagation and contention delays. Critically, they do not adapt well to changes and are impractical for use in large-scale WMNs.

Motivated by their limitations, this chapter proposes a simple distributed algorithm called Period Controlled Pseudo-TDMA (PCP-TDMA) to derive the MAX-CUT in order to determine the shortest superframe for any MTR networks. Specifically, given an initial superframe, PCP-TDMA requires nodes to gradually change the transmission slot of their incident links. The aim is is to reduce the current superframe length. This is achieved by moving all transmission slots closer to the start of the current superframe and thereby the next superframe can start earlier. From simulation results, PCP-TDMA achieves similar performance as the centralized algorithm ALGO-2 [23] in all tested scenarios. As compared to past distributed solutions, e.g., JazzyMAC [26] and ROMA [27], PCP-TDMA provides three advantages over prior methods. First, each node only communicates with its one-hop

neighbors, and it does not require any global topological information. Second, it achieves high fairness because each link is guaranteed to activate at least once in a superframe. Third, the superframe generated by PCP-TDMA is shorter than the other distributed solutions.

The remainder of this chapter has the following structure. Section 4.1 introduces the network model and formalizes the problem. Then Section 4.2 describes the design of PCP-TDMA. The characteristics of the proposed algorithm are analyzed in Section 4.3. Its evaluation is presented in Section 4.4 and Section 4.5 concludes this chapter.

4.1 Preliminaries

This section models a WMN as a connected graph $G(V, E)$ with $|V|$ vertices and $|E|$ edges. Each node $v \in V$ represents a static wireless mesh node, and each edge, denoted as e_{uv} or (u, v), in E corresponds to a directed link from node u to v in G if and only if the Euclidean distance between u and v is smaller than or equal to the transmission range r. Here, each link is supported by a radio and each node u has $b_u \geq |N(u)|$ radios, where $N(u)$ contains the neighbors of node u.

The packets to be transmitted are of equal and unit length. Time is divided into slots of equal length, which are sized accordingly to transmit one packet. Nodes are assumed to be synchronized [98]; e.g., using GPS. The superframe is denoted as $\mathcal{SF}$ and consists of up to P edge sets, where P is the superframe length; aka the period. Define the i-th edge set in $\mathcal{SF}$ as ϵ_i, which contains transmitting links that adhere to the no Mix-Tx-Rx constraint; note, ϵ_i can be empty. Hence, a superframe is defined as $\mathcal{SF} = \{\epsilon_i \mid i \in \{1, \ldots, P\}\}$. In each superframe, every time slot s is numbered sequentially, whereby s_i represents the i-th slot with $i \in \{1, \ldots, P\}$. In addition, the x-th superframe is indexed as $\mathcal{SF}_x$.

To avoid interference, nodes need to know the slots that are used by their neighbors for transmitting and receiving packets. To this end, each node v maintains two

sets: $\mathbf{Tslot}_v$ and $\mathbf{Rslot}_v$; see Section 4.2.1 for details on constructing these sets. As an example, consider $\mathbf{Tslot}_A = \{s_i, s_j\}$. This means node A transmits in slot s_i and s_j. Assume each node knows the $\mathbf{Tslot}_v$ and $\mathbf{Rslot}_v$ of every neighbor v. This is reasonable because each node can include this information in all transmitted data packets. As a result, when selecting a transmitting slot for its link, say (A, B), node A can only choose from the set of *feasible slots*, which is defined as $\mathcal{S}_{(A,B)} = \{s_1, \ldots, s_P\} \setminus (\mathbf{Rslot}_A \cup \mathbf{Tslot}_B)$. In words, slots used for reception and those used by node B for transmission are excluded. Table 4.1 summarizes key notations used throughout this chapter.

Notation	*Description*
G	A directed graph
V	A set of nodes or vertices in G
E	A set of directed links or edges in G
e_{AB} or (A, B)	A link with source node A and destination node B in E
$N(u)$	A set containing node u's neighbors
$\mathcal{SF}_x$	The x-th superframe
P	The length of $\mathcal{SF}$, *aka* the period or superframe length
s_i	The i-th slot in $\mathcal{SF}$
$\mathbf{Tslot}_A/\mathbf{Rslot}_A$	A set that contains node A's transmitting/receiving slots
ρ	The probability that a node attempts to reserve a slot again
T_O	The duration of a timeout timer started by *parent* node
$\mathcal{S}_{(A,B)}$	A set containing link (A, B)'s *feasible slots*
D_{max}	The maximum node degree among all nodes
$\varnothing$	The network diameter

Table 4.1: Key notations in Chapter 4

To define the problem, the aim is to derive the shortest possible superframe; i.e., the smallest P, in a distributed manner. Specifically, given an initial P value, and nodes with MTR capability, design a distributed algorithm that iteratively reduces the superframe length or P value over time. Note, how P is determined initially and adjusted will be discussed in Section 4.4.4 and 4.2, respectively.

4.2 Period Controlled Pseudo-TDMA

The basic idea is as follows. The initial superframe $\mathcal{SF}_1$ has period P. All nodes attempt to reserve a random slot for each of their links. If a node reserves a slot successfully, it will use the slot for data transmission in the next superframe. Otherwise, if there is a collision, a node will attempt to reserve another random slot in the next superframe. After reserving a slot, say s_i, the node will then attempt to improve its current slot s_i by reserving an earlier slot s_j, where $j < i$, in the next superframe. This means if the last slot reserved by nodes is s_c, where $c < P$, then the superframe can be reduced to c; i.e., P is updated to c.

Consider Figure 4.1. Assume $|\mathcal{SF}_1| = P = 6$. In the first P slots, none of the links are scheduled; i.e., $\mathcal{SF}_1$ consists of six empty edge sets. Nodes send a RESV message for each of their links; this is indicated by the gray boxes. Assume the RESV message of all links, except e_{CB}, is delivered successfully. Transmission on link e_{CB} fails because B's reception is affected by the transmission on link e_{BA}. From the next superframe onwards, the transmitter of these links will start to transmit data packets, shown as white boxes, in their reserved slot. This means these links are included in the edge sets of $\mathcal{SF}_2$. Additionally, as soon as a node, say v, successfully reserves a slot, it marks this slot as its transmitting slot and includes this slot into $\mathbf{Tslot}_v$. On the other hand, if a node, say w, receives a RESV message in slot k, it includes k into $\mathbf{Rslot}_w$.

As mentioned earlier, node B fails to receive the RESV message over link e_{CB}. Consequently, in superframe $\mathcal{SF}_2$, node C sends another RESV message in a random slot in the set $\mathcal{S}_{(C,B)}$, say slot s_1, for the unscheduled link e_{CB}. As link e_{CB} does not interfere with any link in the set ϵ_1, it is thus included in ϵ_1 of superframe $\mathcal{SF}_3$.

Nodes can improve the slot of the reserved links by contending for an earlier slot. Continuing the previous example, we see that link e_{BA} occupies slot s_6. In order to improve s_6, node B sends a RESV message for slot s_2; note, this slot is chosen randomly from the set $\mathcal{S}_{(B,A)}$. As a result, link e_{BA} is removed from the edge set ϵ_6

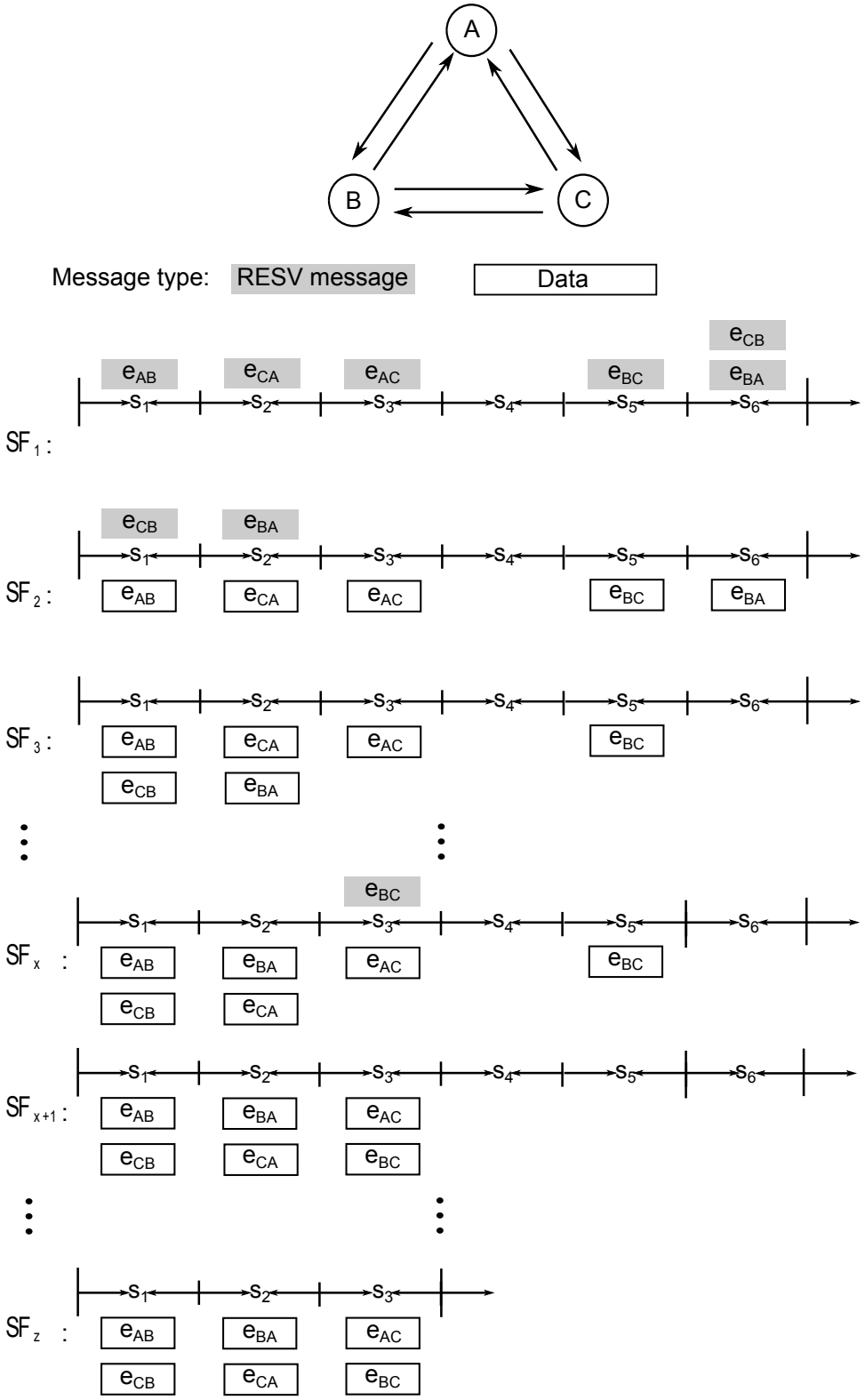


Figure 4.1: An example of PCP-TDMA

and added into the edge set ϵ_2 in superframe $\mathcal{SF}_3$. In addition, nodes B and A will update their set $\mathbf{Tslot}_B$ and $\mathbf{Rslot}_A$, respectively, by replacing element s_6 with s_2. Next, in $\mathcal{SF}_x$, node B tries again to reserve an earlier slot, i.e., s_3, for link e_{BC}. We see that e_{BC} is then scheduled in s_3 of superframe $\mathcal{SF}_{x+1}$.

The next key idea of PCP-TDMA is to reduce the superframe length or P iteratively. Assume node B has the highest node ID among the three nodes, as node B has reached a state where no link can be shifted to an earlier slot without causing interference, node B is prompted to propose a new period. Assume nodes know the slots reserved by its neighbors, then node B searches for the latest time slot used by itself and its neighbors; i.e., s_3 of $\mathcal{SF}_{x+1}$ is used by link e_{BC}. Thus node B proposes $P = 3$. If all nodes approve this new period, meaning slots after s_3 are idle, all nodes set P to three.

Finally, we see that the resulting schedule converges to $\mathcal{SF}_z$ with a length of three. Formally, there is the following definition for the *converged state*:

Definition 1. The system reaches the *converged state* when all nodes are unable to improve the currently reserved slot of all their links.

Referring to Definition 1, the superframe $\mathcal{SF}_z$ and all subsequent superframes will have a period of three. Note, if the topology changes after the period update, e.g., a new node joins or dies, then a new superframe will have to be regenerated. However, this situation does not happen frequently as nodes in a WMN are primarily static. Having said that, Section 4.2.3 will discuss how nodes readjust the superframe when the topology changes.

In the foregone example, we see that PCP-TDMA needs to address the following four sub-problems. Firstly, nodes need to improve their reserved slots over time. Secondly, it is necessary for nodes to reduce the probability of collisions when reserving a random slot. Thirdly, nodes need to determine the last reserved slot. Fourthly, it is important that *all* nodes update their period to the same value and reduce to the shortest possible P.

PCP-TDMA consists of two parts: slot reservation and period minimization. In the first part, nodes send RESV messages to move the activation time of their links nearer to the start of each superframe in order to fully utilize earlier time slots of a superframe. In the second part, nodes communicate with their neighbors to inform each other of idle slots located at the end of a superframe, and then remove these idle slots to shorten the period P.

4.2.1 Part-1: Slot Reservation

The aim is to improve current transmission slots by attempting to reserve earlier slots. Figure 4.2 shows the state diagram of the slot reservation process. Initially, nodes are in the "*Start*" state. Assume link e_{AB} of node A currently has slot s_i. Node A then moves to the state "*Transmit RESV*" and attempts to reserve a random slot s_j in $\mathcal{S}_{(A,B)}$, where $j < i$, by sending a RESV message to node B in slot s_j. The RESV message is sent with a probability of $\rho = \frac{i}{P}$; recall that i is the slot index number, and P is the current superframe length. Observe that ρ is biased towards links with a bigger slot number; i.e., those near the end of the current superframe will have a higher priority to move to an earlier non-conflicting slot.

A node, say B, that receives a RESV message moves into the state "*Receive RESV*". Assume node B receives a RESV message without any conflict. It then replies immediately with a grant or GRT message. Node B then updates **Rslot** to record slot s_j as its receiving slot; i.e., it replaces slot s_i in $\mathbf{Rslot}_B$ with s_j. After that node B goes back to the "*Start*" state. When node A receives the GRT message from B, it moves to the "*Update* ***Tslot***" state to mark slot s_j as its transmitting slot by replacing slot s_i with s_j in the set $\mathbf{Tslot}_A$. It then moves back to the "*Start*" state.

If node B experiences a collision, i.e., it did not receive the RESV message from A, then there will be no GRT message. In this case, node A concludes that the reservation has failed. It thus retains the current transmitting slot s_i for link e_{AB}.

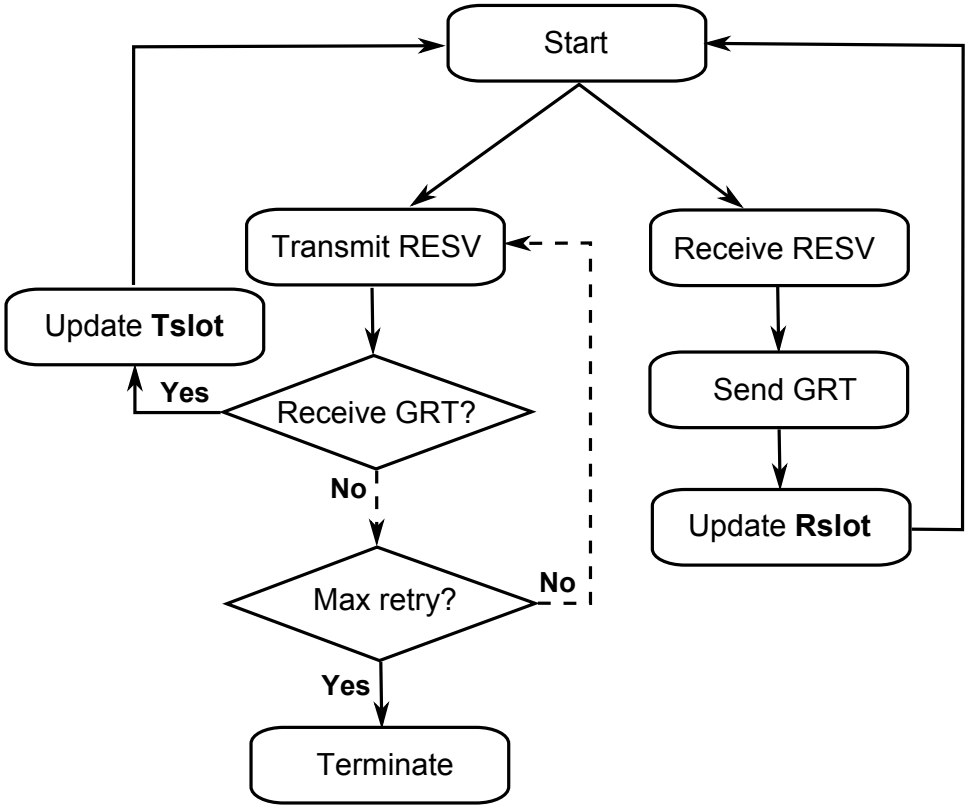


Figure 4.2: State diagram for PCP-TDMA's slot reservation process

Node A will either go back to the "*Transmit RESV*" state to retransmit a RESV message with probability ρ in a random slot from $\mathcal{S}_{(A,B)}$ in the next superframe, or go to the "*Terminate*" state. The state node A chooses depends on whether it has tried to transmit a RESV message for a given maximum retry threshold. The retry limit is set to $|\mathcal{S}_{(A,B)} \setminus \{s_i, \ldots, s_P\}|$, where s_i is the current reserved slot. This allows nodes to try to reserve in every earlier slot in $\mathcal{S}_{(A,B)}$ before it terminates the slot reservation process. Once in the "*Terminate*" state, a node no longer tries to move its current slots.

4.2.2 Part-2: Period Minimization

This part consists of two stages: new P proposal and its confirmation. The aim is for nodes to learn the shortest feasible period and to update their current period. Eventually, all nodes in the network will use the same shortest P, and the superframe period can no longer be shortened.

4.2.2.1 Stage-1: New P Proposal

This section explains how nodes propose a new period. To reduce signaling overheads, only nodes with the highest ID among all their neighbors have the right to propose. Assume that node A is such a node. After reaching the "*Terminate*" state in Part-1, it searches for the largest slot s_k that is occupied by a transmitting link. Formally,

$$P' = \underset{k \in \{1,\ldots,P\}}{\arg\max} (s_k \cap \{\mathbf{Tslot}_u \cup \mathbf{Rslot}_u\} \neq \emptyset) \tag{4.1}$$

where $u \in \{N(A) \cup A\}$. Node A compares P' against the current period P. If $P' < P$, then node A becomes the root node. It sends a PROP$\{A, P'\}$ message to its neighbors. The message includes its ID and the proposed period P'.

In the sequel, the following definition of *parent* and *child* is needed. A *parent* of a node A is defined as the neighbor that has transmitted a PROP message to A. All other nodes in $N(A)$ are known as the *children* of node A. For each PROP message, a node will keep a separate record of the corresponding parent and child nodes. In addition, after transmitting a PROP to every child, a node starts a timer called T_O. The duration of T_O is a design parameter that can be changed according to traffic requirements or network topology.

When a node, say C, receives a PROP $\{A, P'\}$ message from its neighbor B, node C needs to determine whether to accept or reject the proposed period P'. This process is illustrated by the state diagram shown in Figure 4.3. Upon receiving a PROP message, node C will record neighbor B as a *parent*. Then node C needs to determine whether it is a duplicated PROP message. To do this, node C checks the following two elements contained in the PROP message: ID and P'. If the ID of the received PROP message matches the ID contained in a previously received PROP message, and these two PROP messages have the same P' value, then the newly received PROP message is a duplicate. Node C discards the duplicated PROP message and will not reply to parent B.

On the other hand, if the PROP message is new, then C will determine the

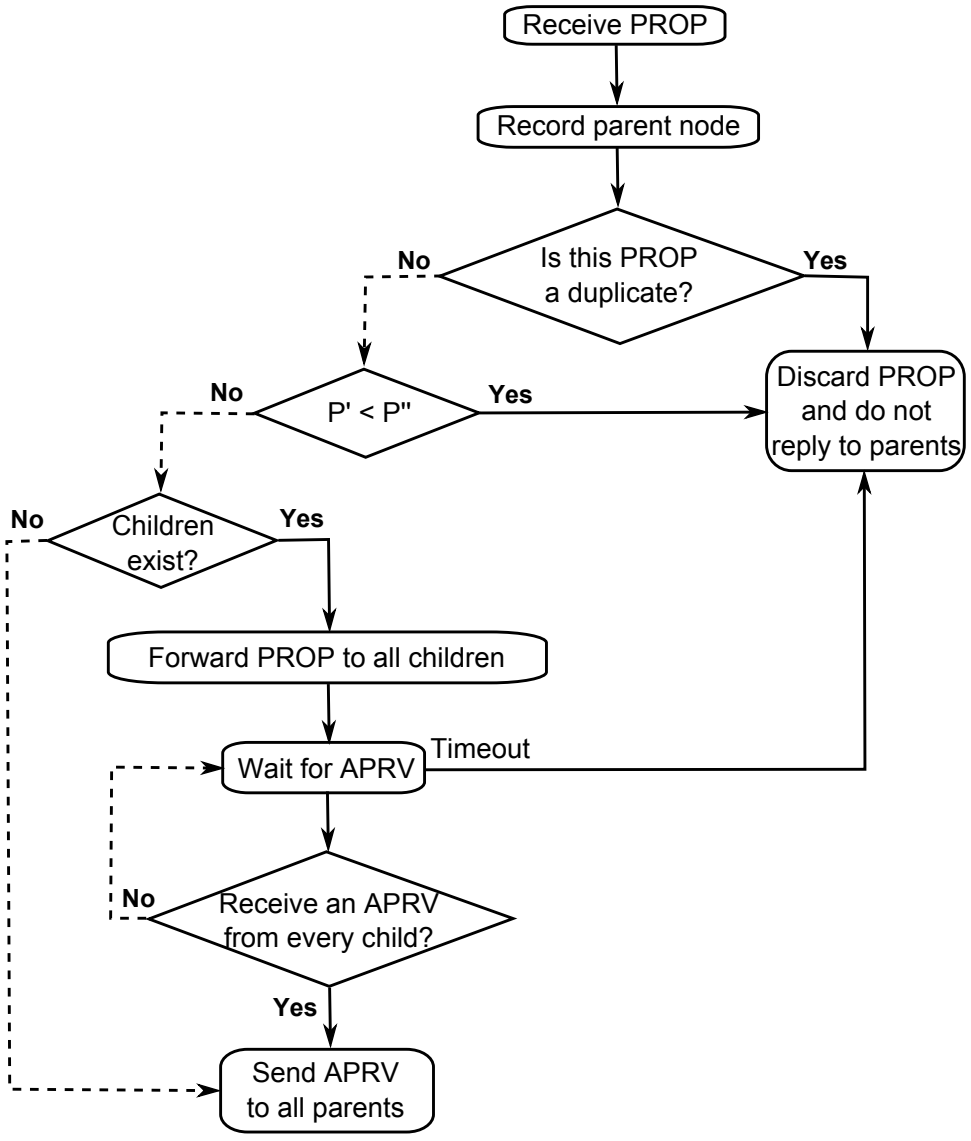


Figure 4.3: The propagation of a PROP message

validity of the proposed period P' as follows. Node C first finds the largest occupied time slot $s_{k'}$. That is,

$$P'' = \underset{k' \in \{1,\ldots,P\}}{\arg\max} \left(s_{k'} \cap \{\mathbf{Tslot}_u \cup \mathbf{Rslot}_u\} \neq \emptyset\right) \tag{4.2}$$

where $u \in \{N(C) \cup C\}$. Node C compares P' against P'' to determine whether the proposed period P' can be accepted by node C. If P' is smaller than P'', meaning P' cannot be node C's new period, then node C discards the message PROP $\{A, P'\}$ and does not reply to any *parent*. On the contrary, if $P' \geq P''$, node C approves P' and forwards the PROP message to all its *children*, if there are any. Node C then enters the state "*Wait for APRV*", where C expects all its *children* to send a message APRV$\{A, P'\}$ back as an approval of the proposed period P'. If C does not receive an APRV message from every child within a duration of T_O, a Timeout event occurs. This causes node C to discard the message PROP $\{A, P'\}$ and not to reply with an APRV message. However, if C collected every APRV before T_O, then node C goes to the last state "*Send APRV to all parents*".

4.2.2.2 Stage-2: New P Confirmation

This stage starts when a root node, say A, has successfully collected an APRV message from all its neighbors (or children) in Stage-1. The aim of this stage is to inform all nodes the approved period and the start time of the new superframe. The key challenge is to have all nodes start the new superframe at the same time.

In this stage, any node, say C, uses a message called UPDATE $\{A, P', t, \tau_C\}$ to inform its children that the root node A is going to start a superframe with period P'. Here, the element t is a time stamp (e.g., unix epoch timestamp) of when this UPDATE message is generated by the root node, and τ_C indicates the time slot that node C begins using the new P' instead of the current period P. Here, all four elements are important because they are also used to guarantee the uniqueness of each UPDATE message.

To explain how node C calculates its starting slot τ_C, assume node C sends the UPDATE $\{A, P', t, \tau_C\}$ message to its children in superframe $\mathcal{SF}_y$. The value of τ_C must satisfy the following two requirements.

1. **Requirement-1**: τ_C must be a slot in $\mathcal{SF}_{y+2}$. This is because node C requires two superframes, namely $\mathcal{SF}_y$ and $\mathcal{SF}_{y+1}$, to send an UPDATE message and receive an ACK from all its children.

2. **Requirement-2**: τ_C must be $n \times P'$ slots after the starting slot of C's parent, where $n \in \mathbb{N}$. This is to ensure that C starts the new superframe with period P' simultaneously with its parent.

Here, the new superframe $\mathcal{SF}_{y+2} = \{\epsilon_i \mid i \in \{1, \ldots, P'\}\}$, where each edge set ϵ_i in $\mathcal{SF}_{y+2}$ is equal to its corresponding edge set in $\mathcal{SF}_y$. Note that each edge set ϵ_i, for $i > P'$, in $\mathcal{SF}_y$ is empty and thus is not considered.

Figure 4.4 is used to explain how a new period is confirmed and updated by every node. Firstly, a root node, say A, will carry out the steps on the left branch. It sends the message UPDATE$\{A, P', t, \tau_A\}$ to all its children and waits for their ACK. At time slot τ_A, if A has received an ACK from every child, node A goes to the last state in the left branch; i.e., "*Start new superframe with period of* P' *in slot* τ_A". Otherwise, node A returns to the sending UPDATE state at the beginning of the left branch after "*Recalculate* τ_A". Here, the starting slot τ_A is recalculated as the first slot after two superframes; cf. Requirement-1.

Next, Figure 4.4 is used to describe how a node confirms and updates the new period P' when it receives an UPDATE message. According to the right branch of Figure 4.4, if a node, say C, receives an UPDATE $\{A, P', t, \tau_B\}$ message from its parent B, node C acquires the following information: parent B is going to start a new superframe with period P' from slot τ_B onwards.

First, node C compares P' against the current period P. If P' is bigger than P; i.e., $P' > P$, node C does not send an ACK to its parent and it will not change its period. Otherwise, if $P' < P$, node C goes through the remaining states in the right

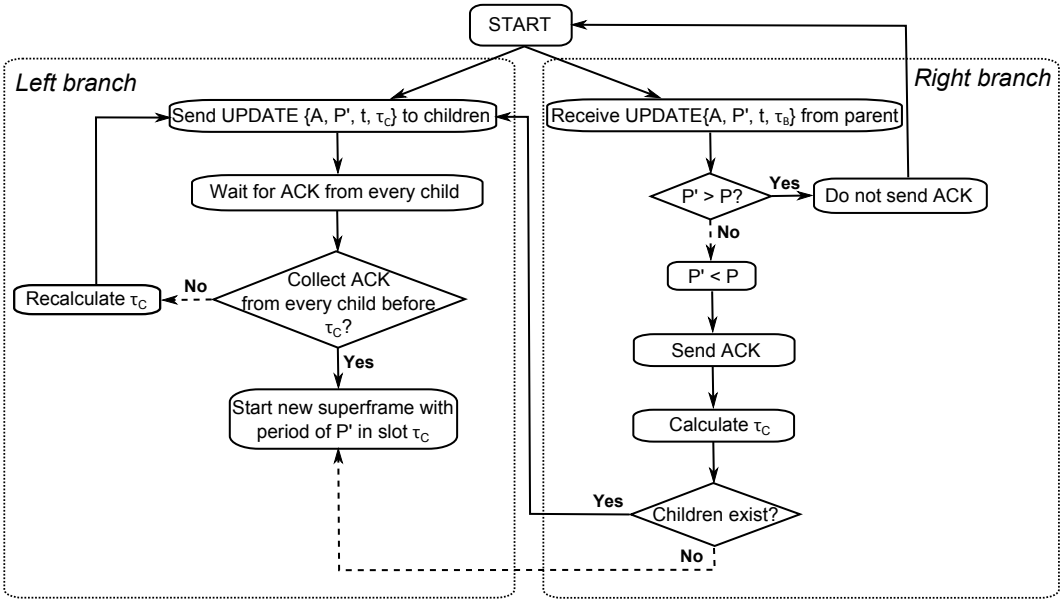


Figure 4.4: Confirmation of a new period

branch. It first responds to parent B with an ACK to inform B that the period update event has been noted. Then node C will calculate its own starting slot τ_C for the new period. If node C has children, it moves to the left branch to inform its children. Otherwise, if C has no children, it goes to the last state in the left branch where C starts the new superframe with period of P' from τ_C onwards. With the propagation of the UPDATE messages, the period update event occurs at each node in the network. Finally, all nodes conform to the same period P'.

One question that arises is what if node C receives an UPDATE message whose P' value is equal to the current period P. Although the superframe length does not change, a different starting slot contained in the newly received UPDATE message leads to a different starting slot τ_C for node C. Thus, to ensure that every node starts the new superframe simultaneously, node C will check the time stamp t of this UPDATE message. Nodes will adopt this UPDATE message if it is older because this means it has existed for a longer time period and thus, covers more nodes. In the event that the time stamp is the same, nodes will accept the UPDATE if it contains a higher root node ID. Otherwise, the UPDATE message will be discarded.

4.2.3 Topological change

Whenever new nodes join, the current period may need to be re-adjusted to ensure that new links can be scheduled without interference. In addition, the algorithm also needs to make sure existing links remain unaffected by new links.

Assume there is a new node F. Let E be its neighbor that is already connected to the network. Node E will transmit its current schedule and current period P to node F in a random slot, say s_r, in $\mathbf{Tslot}_E$. Node F will then record s_r in $\mathbf{Rslot}_F$. Upon receiving node E's schedule, node F inspects the schedule of E, and sends a RESV message to node E in a random *feasible* slot s_t in $\mathcal{S}_{(F,E)}$. If node F receives a GRT message from E, then the slot reservation is successful. Node F and E add the reserved slot s_t into the set $\mathbf{Tslot}_F$ and $\mathbf{Rslot}_E$, respectively.

In the case where no *feasible* slots are available, meaning every slot has a conflict with node F's outgoing links, then node F needs to expand the current superframe by one slot. Specifically, the new superframe needs to have a period of $P+1$, where the edge sets $\{\epsilon_i | i \in \{1, \ldots, P\}$ in the new superframe are equal to that of the current superframe, and the one extra edge set ϵ_{P+1} contains all F's unscheduled links.

To expand the superframe, node F sends a JOIN$\{P+1,\ \tau_0\}$ message to an existing neighbor node at random; let it be node E. It then becomes a root node and sends a EXP$\{E,\ P+1,\ t,\ \tau_E\}$ message to its children, where E is the root node ID, $P+1$ is the new period, t is the time stamp and τ_E is the starting slot of the new superframe. This message is propagated to all other nodes in the network; see Figure 4.5. Observe that the depicted process is similar to how an UPDATE message is processed in Section 4.2.2.2, except that nodes do not have to verify the validity of the proposed period $P+1$.

On the other hand, if existing nodes leave the network, two cases are taking into consideration. In the first case, the leaving node, say Q, is the root node which established the current schedule. It means that before Q left, its incident links

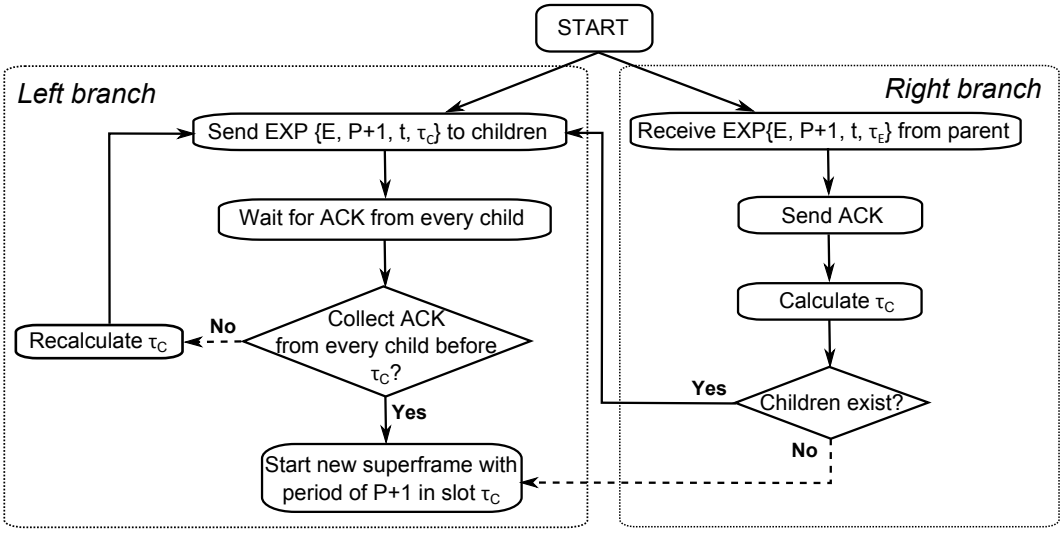


Figure 4.5: Re-adjusting the period when a new node joins

occupied the largest slot s_P of the current period P. Thus the neighbor(s) of node Q propagate(s) a message to notify all other nodes that the root node Q has left the network. Upon receiving such a message, the remaining nodes will initiate Part-2 to propose a new period P'. However, if Q is not a root node, then no action needs to be taken because the largest occupied slot remains the same, and thus the length of the current period is not affected.

4.3 Analysis

This section analyzes several properties of PCP-TDMA, including the configuration of the initial period, the correctness of the schedule, the self-stabilizing feature of the algorithm, and the time required for Part-2 of PCP-TDMA to finish.

Proposition 1. *Given an arbitrary topology, with a maximum node degree D_{max}, setting the initial period P_i to at least $2 \times D_{max}$ guarantees each link will reserve a slot.*

Proof. Consider a link (A, B). To schedule this link without interference, the following inequality must be true:

$$\mathcal{S}_{(A,B)} \neq \emptyset \tag{4.3}$$

Equivalently,

$$\{s_1, \ldots, s_P\} \setminus (\mathbf{Rslot}_A \cup \mathbf{Tslot}_B) \neq \emptyset \tag{4.4}$$

This indicates that the number of feasible slots must be greater than zero. Since the values are all integers, there is the following inequality,

$$P - |\mathbf{Rslot}_A \cup \mathbf{Tslot}_B| \geq 1 \tag{4.5}$$

If both node A and B have D_{max} neighbors, that means A has D_{max} incoming links, and B has D_{max} outgoing links. In the worst case, all these said links are scheduled in a distinct time slot. Consequently, there is $|\mathbf{Rslot}_A| = |\mathbf{Tslot}_B| = D_{max}$. Note, link (B, A) is counted twice. Thus, there is

$$P - (2 \times D_{max} - 1) \geq 1 \tag{4.6}$$

Hence, $P \geq 2 \times D_{max}$ is obtained to ensure that all links have at least one feasible slot to reserve, which proves the proposition. □

Proposition 2. *PCP-TDMA produces an interference-free schedule.*

Proof. Here, only Part-1 (**Slot Reservation**) of PCP-TDMA is worth analyzing because Part-2 (**Period Minimization**) reduces the length of the superframe without changing the link schedule. In Part-1, consider a node A and assume links e_{AB} and e_{xA} have reserved the same slot s_i. The following two cases are considered:

Case 1: Node A transmits a RESV message in slot s_i even though a link e_{xA} exist. Recall that A can only select a transmitting slot from the set $\mathcal{S}_{(A,B)}$. The fact that slot s_i is in both $\mathcal{S}_{(A,B)}$ and $\mathbf{Rslot}_A$ contradicts the definition of a *feasible slots* set, whereby $\mathcal{S}_{(A,B)} = \{s_1, \ldots, s_P\} \setminus (\mathbf{Rslot}_A \cup \mathbf{Tslot}_B)$.

Case 2: Node A and a neighbor B choose to send a RESV message in time slot s_i. As the reservation is successful, this means node A receives the RESV message from node B while A is sending its RESV message to B. This contradicts the no Mix-Tx-Rx constraint.

In both cases, PCP-TDMA does not generate a schedule with interference, which proves the proposition. □

Proposition 3. *PCP-TDMA has the property of self-stabilization, which ensures all nodes in the network to end up in a correct state; i.e. the* converged state.

Proof. This proof shows that nodes using PCP-TDMA reach the *converged state*. In Part-1, a node, say A, iteratively attempts to replace the current reserved slot s_i with a random slot s_j from the set $\mathcal{S}_{(A,B)}$ for its link e_{AB}, where $j < i$. If the attempt is successful, the maximum retry limit is updated to $|\mathcal{S}_{(A,B)} \setminus \{s_j, \ldots, s_P\}|$. Otherwise, if this attempt fails, the retry limit becomes $|\mathcal{S}_{(A,B)} \setminus \{s_i, \ldots, s_P\}| - 1$ because s_j is removed from $\mathcal{S}_{(A,B)}$. Thus, the max retry threshold is guaranteed to decrease to zero at some time, meaning node A will eventually move to the "*Terminate*" state. Note, this "*Terminate*" state is equivalent to the *converged state* because nodes no longer change their transmitting and receiving slots. Therefore, PCP-TDMA is self-stabilizing because all nodes are guaranteed to reach the *converged state*. □

Proposition 4. *The number of slots, denoted as σ, required by Part-2 of PCP-TDMA in an arbitrary network with diameter $\varnothing$ is bounded by $2 \times P \leq \sigma \leq 4 \times \varnothing \times P$.*

Proof. First consider Stage-1 of Part-2 and bound of the number of superframes a root node requires to receive an APRV from every neighbor after initiating a PROP message. In the best case, this can be done in only one superframe if the transmission of PROP happens successively from root node to the farthest node, and from parents to children. Then within the same superframe, after a PROP message is received by the farthest node, the transmission of APRV messages occurs in the exact opposite sequence of PROP's transmission, i.e., from the farthest node to root node, children to parents. However, without this specific transmission order, it may take up to at most $\varnothing \times P$ slots to propagate a PROP message to the farthest node from the root node and another $\varnothing \times P$ slots for the root node to collect all APRV messages. This happens when the hop-distance between root and the farthest node equals the network diameter $\varnothing$. Thus, the number of time slots PCP-TDMA takes to perform

Stage-1 of Part-2 is at least P, and at most $2 \times \varnothing \times P$. Similarly, the algorithm needs the same results for Stage-2 of Part-2 for the transmission of UPDATE and ACK messages. Therefore, the number of slots required by Part-2 is bounded by $[2 \times P, 4 \times \varnothing \times P]$. □

4.4 Evaluation

This section evaluates the performance of PCP-TDMA using MatGraph [97], a Matlab toolkit that works with simple graphs. Each node is assumed to have a dedicated antenna for every neighbor. Experiments are conducted over bipartite or random topologies. For experiments that use bipartite graphs, a linear and a grid network consisting of 16 nodes are constructed. For random topologies, 50 nodes are randomly placed on a $100m \times 100m$ square area in order to study the impact of two parameters: node degree and transmission range. The degree of each node varies from 5 to 15. The transmission range of nodes varies from 30m to 100m.

The proposed algorithm PCP-TDMA is compared against ALGO-2 [23], a centralized MTR link scheduler, and two distributed algorithms JazzyMAC [26], and ROMA [27]. The aim of ALGO-2 is to generate a bipartite graph with maximal matching by placing nodes into two sets: Set1 and Set2. Initially, all nodes are included in Set1 while Set2 is empty. ALGO-2 then moves a node from Set1 to Set2 if doing so increases the number of active links. After processing all nodes, a max-cut is derived. In time slot i, nodes in Set1 transmit to nodes in Set2. Then, upon removing all activated links from nodes in Set1 to those in Set2 from the network, the above process is repeated on the revised topology to generate the next max-cut. ALGO-2 terminates when it has scheduled all links. The superframe length is equal to the total number of max-cuts obtained by ALGO-2. This is the minimal superframe length that ensures every link is activated at least once.

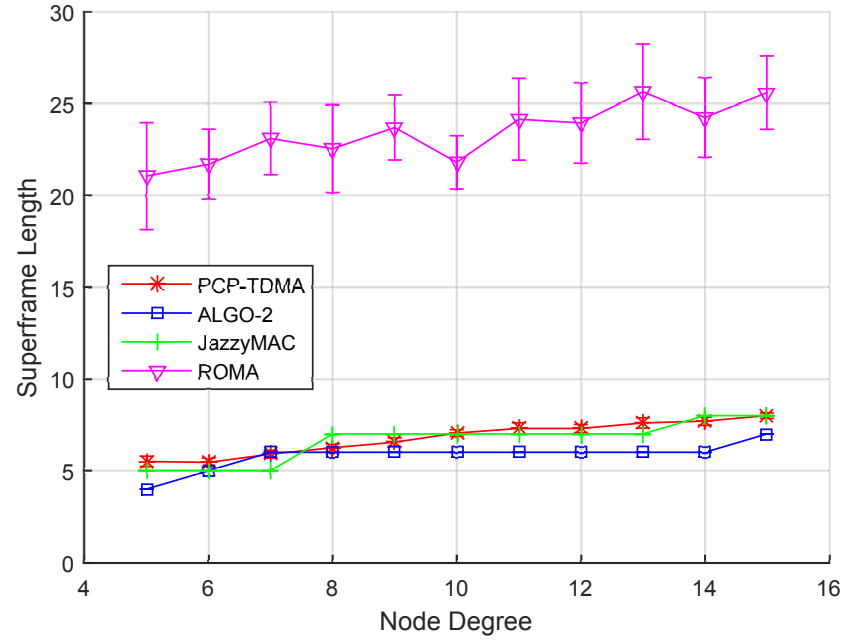
JazzyMAC initially assigns tokens to nodes according to a centralized scheme; i.e., graph coloring. A node becomes a transmitter when it holds the token of all

its incident links. When a node finishes its transmission, it passes the token to the other end of the link. ROMA is a distributed scheme where nodes are synchronized and uses two-hop topology information to compute a schedule. ROMA evenly and randomly splits nodes into transmitters and receivers in each slot, which are paired together for data transmission. Then ROMA solves any contention according to the priority of each node, where the priority is calculated based on node ID. The node with the highest priority among contending neighbors has the right to transmit without conflicts in that time slot.

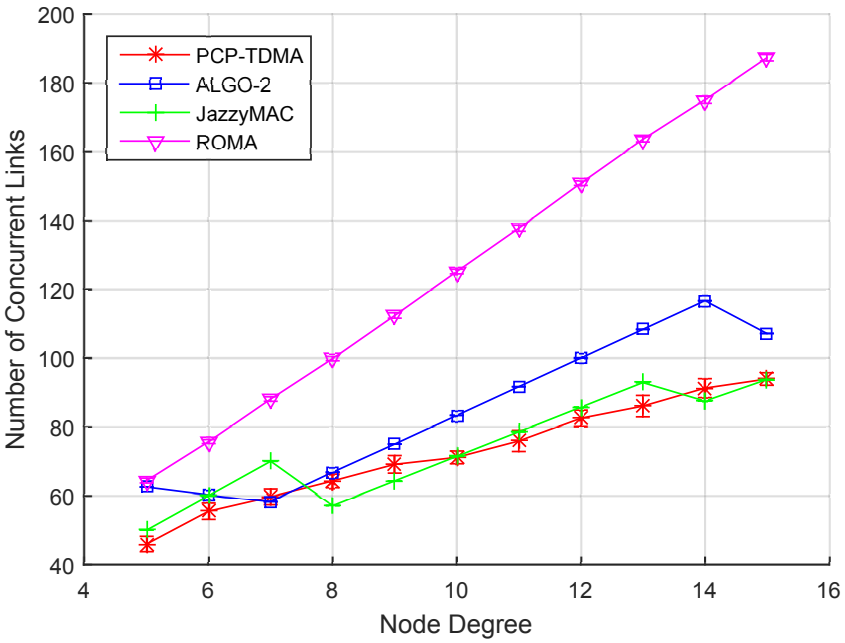
The experiments compare metrics such as superframe length and the number of concurrent active links. In addition, the experiment also measures the number of time slots and signaling messages required for PCP-TDMA to reach convergence. All presented results are an average of 20 simulation runs; each with a different topology. The error bars shown in the line graphs indicate 95% confidence interval of the mean value.

4.4.1 Node Degree

Figure 4.6 (a) shows the superframe length calculated when nodes have 5 to 15 neighbors. We can see that all the algorithms generate a relatively short superframe with similar length except ROMA. The superframe of ROMA is approximately two times more than that of other algorithms. This is because ROMA splits all nodes into transmitters and receivers randomly in each time slot. However, the other three algorithms construct a max-cut comprising of unscheduled links, and thus they schedule the maximal number of unscheduled links in each slot which leads to a shorter superframe. Interestingly, when nodes have a degree of seven, observe that JazzyMAC generates a superframe with length that outperforms the centralized algorithm ALGO-2 by one. The reason is because in the initial greedy graph coloring stage of JazzyMAC, it occasionally generates the optimal graph coloring. However, ALGO-2 in these cases fails to derive the optimal max-cut.



(a)



(b)

Figure 4.6: Performance of different algorithms under increasing node degrees. (a) Superframe length. (b) Number of concurrent links.

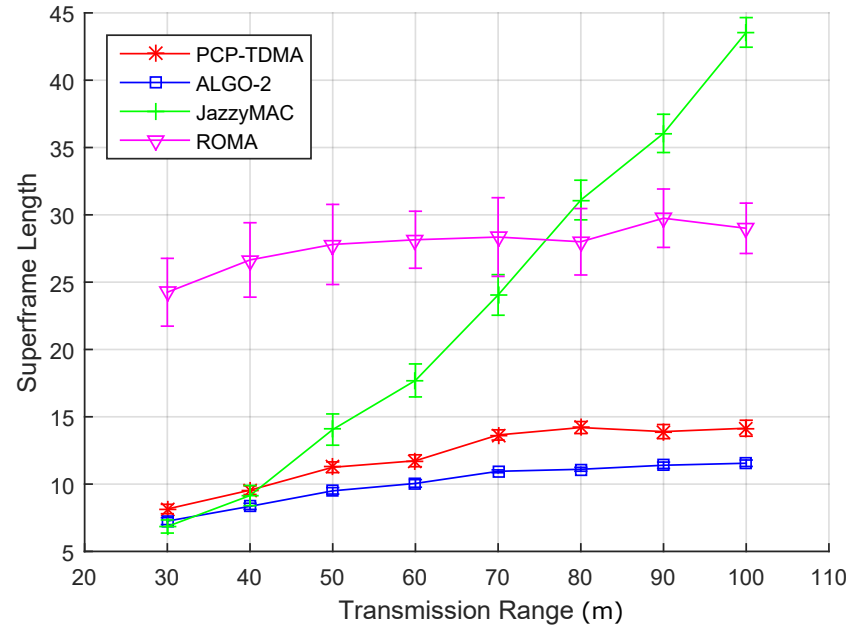
Figure 4.6 (b) shows the average number of concurrent links in each slot with increasing node degrees. When using ROMA, the number of concurrent links increased from 64.4 to 187.3. Specifically, it significantly outperforms other tested algorithms when node degree increases from 6 to 15. This is because ROMA does not remove any links after links are scheduled. As the node degree increases, the number of existing links increases. Hence, ROMA has more chances to repeatedly schedule previously activated links as opportunistic links. However, PCP-TDMA does not schedule opportunistic links because it wants to fill every slot with the most number of unscheduled links. For ALGO-2, scheduled links are intentionally removed from the network. In JazzyMAC, opportunistic links do not exist because of its token scheme. As a result, these three algorithms have poorer performance in terms of the number of activated links in each slot.

Note that for PCP-TDMA, ALGO-2 and JazzyMAC, the product of superframe length and number of concurrent links per slot equals $|E|$. With each increment in node degree, $|E|$ increases by 50. Thus, we can see that if the superframe length does not increase, the number of concurrent links rises linearly. For example, when the node degree increases from seven to 14, the superframe length of ALGO-2 in Figure 4.6 (a) remains at six, while the number of concurrent links of ALGO-2 in Figure 4.6(b) increases linearly. The increment value 8.3 is the result of $\frac{50}{6}$, where 50 is the number of added links, and six is the number of slots in a superframe. Moreover, we notice that when the superframe length increases by one or more, the number of concurrent links decreases. For instance, the superframe length of JazzyMAC increases from five to seven in Figure 4.6 (a) when the node degree increases from seven to eight. With more slots in a superframe, from Figure 4.6(b), there will be fewer links in each slot on average.

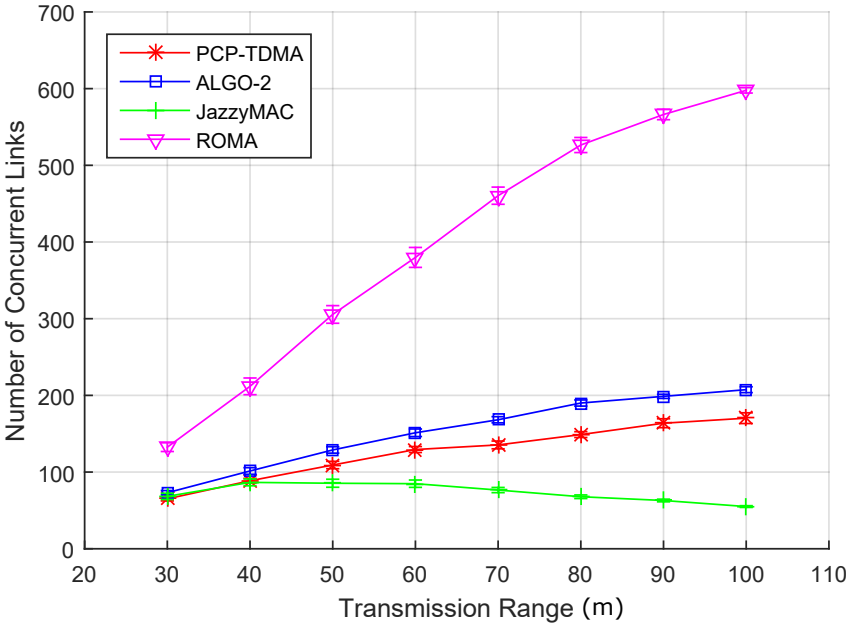
4.4.2 Transmission Range

Figure 4.7 (a) shows that ALGO-2 generates the shortest superframe length, which gradually increases from 6.2 to 11.5. The key reason for this increase is because more links are established between nodes as the transmission range increases. The superframe length of PCP-TDMA is close to that of ALGO-2; i.e., PCP-TDMA produces superframes with at most 3.1 additional slots. For ROMA, its superframe length is fairly high at around 27. This is because ROMA splits all nodes into transmitters and receivers randomly in each time slot. The superframe length of JazzyMAC is similar with ALGO-2 and PCP-TDMA when the transmission range is 30 to 40m. However, from 40m onwards, JazzyMAC shows a sharp increase in superframe length. This is because in JazzyMAC a node is allowed to transmit on all its links only after it has the token of all its links. Consequently, in some cases, time slots are wasted while waiting for tokens to return. Thus, JazzyMAC's performance degrades when nodes need to collect more tokens from more neighbors.

Figure 4.7 (b) compares the average number of concurrent links per time slot. ROMA results in the most concurrent links because of opportunistic links, i.e., links that have transmitted in earlier slots and are added to a slot for the sole purpose of increasing the number of activated links [23]. For ALGO-2 and PCP-TDMA, the capacity can be increased if opportunistic links are added. Therefore, the result does not indicate that ROMA has a higher scheduling efficiency. Further, as shown in Figure 4.7(a), ROMA has a significantly longer superframe. This means some links will have a lower throughput as compared to being scheduled by PCP-TDMA. The number of concurrent links when using ALGO-2 and PCP-TDMA doubles when the transmission range reaches 100m. For longer transmission ranges, the difference between ALGO-2 and PCP-TDMA is at most 20%. For JazzyMAC, the number of concurrent links is reduced by half when the transmission range increases from 40m to 100m. Thus, it is not suitable for random topologies when nodes have many neighbors. Note, at 100m, the network is almost fully connected. Thus, all results



(a)



(b)

Figure 4.7: Performance of different algorithms under increasing transmission range, (a) Superframe length, and (b) Number of concurrent links.

remain the same after 100m.

4.4.3 Bipartite Graphs

Figure 4.8 compares the superframe length generated by different algorithms for bipartite networks such as line and grid topology. We can see that both ALGO-2 and JazzyMAC have the shortest superframe length; i.e., two slots. This indicates that every node is acting as a transmitter in one time slot and as a receiver in the next slot; see Figure 4.9 (a) for an example, where the number next to links indicates the x-th time slot of one superframe. The reason for the shorter superframe is because ALGO-2 constructs max-cuts and JazzyMAC applies optimal graph coloring during bootup. PCP-TDMA yields superframe close to four slots. This is because in PCP-TDMA, links are scheduled in a random order. Take Figure 4.9 (b) as an example. If link e_{AB}, e_{BA}, e_{CD} and e_{DC} are scheduled first as per the indicated slot number, then link e_{CB} and e_{BC} require two additional slots for interference-free transmission. In conclusion, the superframe of PCP-TDMA has an upper bound of four slots for bipartite graphs.

4.4.4 Impact of initial period on convergence time

This section studies how the initial period value P_i, i.e., the length of superframe $\mathcal{SF}_1$, affects the convergence time of PCP-TDMA. To do this, compare the convergence time when the initial period P_i is to three different values. Figure 4.10 illustrates the average number of time slots required for PCP-TDMA to reach convergence when using the following initial periods: P_{i1}, P_{i2} and P_{i3}. Here, P_{i1} is set to $2 \times D_{max}$, P_{i2} to a constant of 10, and $P_{i3} = \lceil D_{max}/3 \rceil + 5$, where D_{max} is the maximum node degree among all nodes. The value of P_{i1} ensures that all links have at least one feasible slot to reserve for general topology. The value of P_{i3} is the most suitable value found experimentally for this particular topology. This simulation is performed on a 50-node network, with node degree increasing from five

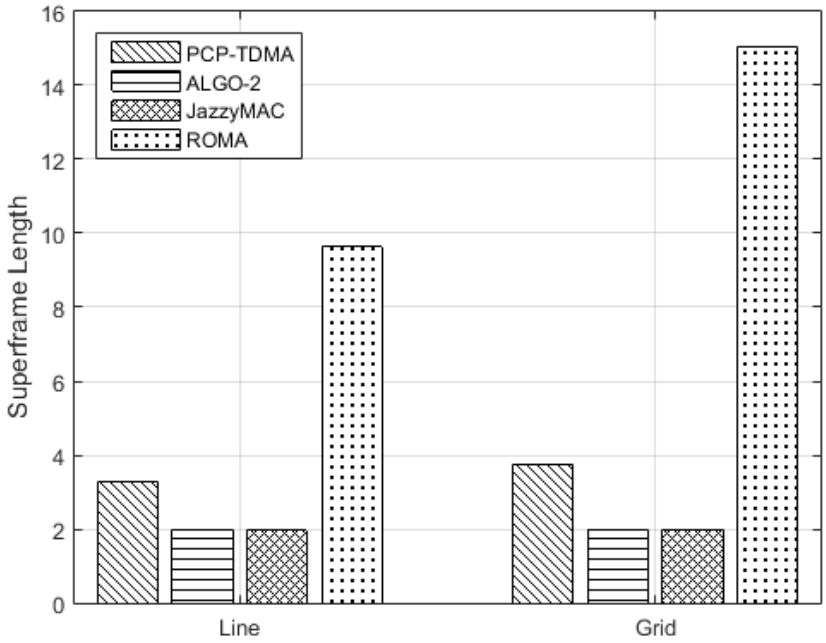


Figure 4.8: Superframe length for bipartite networks

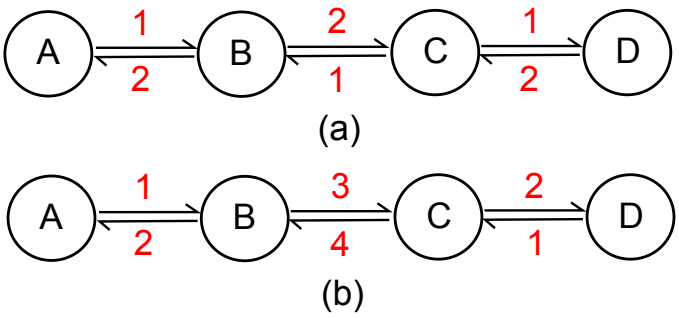


Figure 4.9: Example schedules for a line topology

to 15. Overall, we see a rising trend in convergence time as node degree increases. This is because with increasing number of links, PCP-TDMA requires longer time to schedule every link.

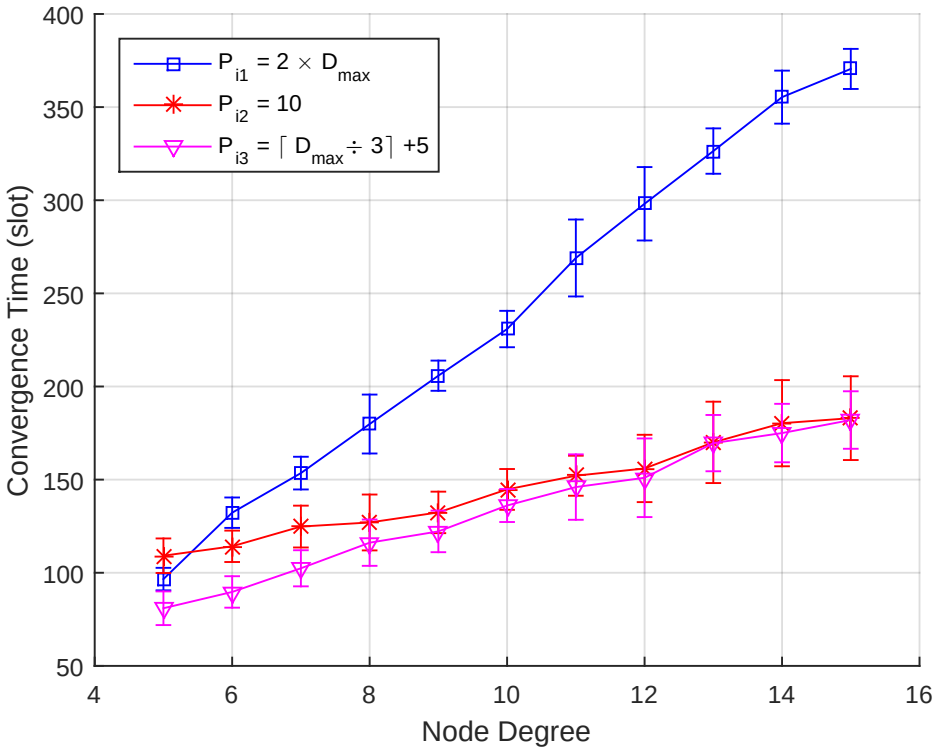


Figure 4.10: Convergence time under increasing node degrees

Figure 4.10 compares the convergence time when using different P_i values. The three curves start around 100 slots. However, the convergence time when using P_{i1} then rises significantly to 370 slots, whereas the number of slots when using P_{i2} and P_{i3} rose steadily to reach just 182. The reason is that, with the increase in node degree, the difference between the P_{i1} and the final period P_f increases rapidly, where P_f is the length of superframes used by nodes when convergence is reached. This means when links are scheduled initially, they tend to be randomly scattered in a longer superframe. Thus nodes require more time to improve their reserved slots repeatedly, in order to reduce the superframe length from P_{i1} to P_f. Using P_{i2} as the initial value results in the minimum increase, about 70 slots. The reason is that when node degree goes up, the difference between P_{i2} and P_f decreases as

P_{i2} is a constant. However, using a fixed integer as P_i is not practical because P_f increases proportionally to the maximum node degree. This means that if P_i is set to a smaller value than P_f, PCP-TDMA can never compute a superframe because interference between links always exists. Thus PCP-TDMA must ensure that P_i is greater than P_f. From these results, the algorithm configures P_i to be P_{i3}, which ensures a relatively small and constant difference from P_f. We can see in Figure 4.10, among the three P_is, the convergence time when using P_{i3} is the shortest, from 80 to 182 time slots.

4.4.5 The number of signaling messages

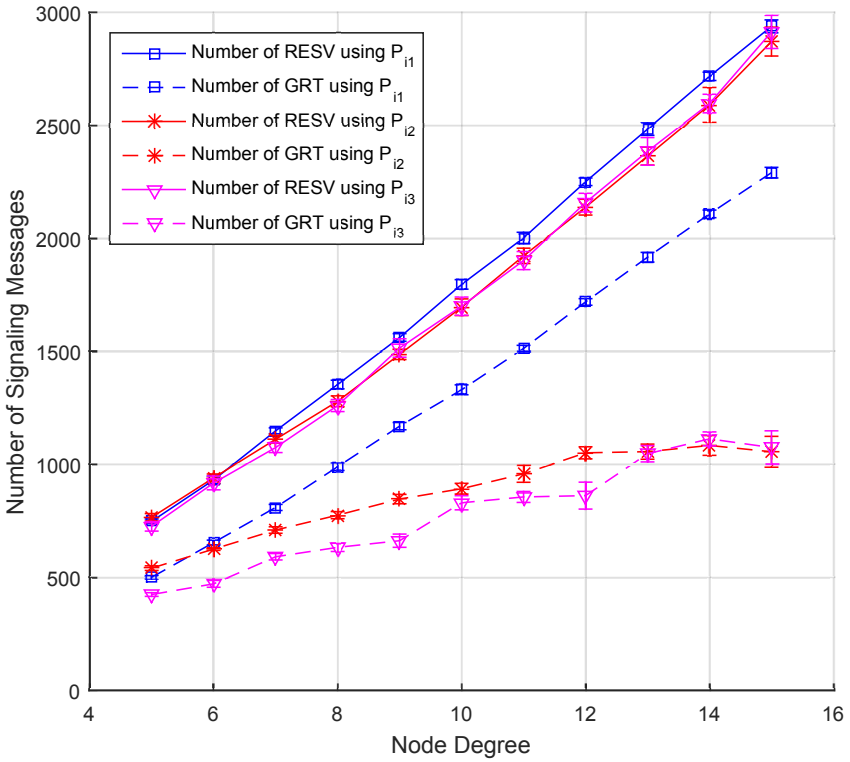


Figure 4.11: Total number of RESV and GRT messages transmitted to reach *converged state*

This section studies the number of signaling messages, including RESV and GRT, used by all nodes to reach *converged state*. The network configuration is the same as Section 4.4.4. Figure 4.11 compares the total number of message exchanges

incurred by PCP-TDMA to stabilize the schedule for all links when using different initial period values with increasing node degree. We see that the number of GRT messages when using P_{i1} is significantly higher than using P_{i2} and P_{i3}; in fact, up to 50% more. This is because P_{i1} is greater than the other two P_i values. With a longer initial superframe, there is a higher successful rate of reserving a random slot. For the same reason, the GRT messages when using P_{i2} is also more than that of $P_{i3}$when the node degree is five to 12. Note, from 13 node degrees onwards, these two curves overlap because $P_{i2} = P_{i3}$ when degree is 13 to 15. In addition, we can see that the number of GRT message when using P_{i3} shows a step shape because P_{i3} is a staircase function.

On the other hand, we see that the numbers of RESV messages when using P_{i1}, P_{i2} and P_{i3} are very close. The reason is that, although nodes using P_{i1} can easily reserve a slot for their links using fewer RESV messages as compared to using P_{i2} and P_{i3}, they require more RESVs to improve reserved slots. Interestingly, the number of RESV messages rises linearly with increasing node degree. This indicates that nodes reserve 3.5 times on average for each of their links to allocate every link in the ideal slot, which is the earliest feasible slot for the particular link. This value does not increase with increasing node degree.

4.5 Conclusion

Advances in radio technologies now allow nodes to have multiple transmit or receive capability over the *same* frequency. To this end, this chapter proposes PCP-TDMA; a MAC that derives a TDMA schedule using only local information. This is significant because it reduces the need to send topological information to a central node/server. Another distinguishing feature is that nodes can start data transmission immediately whilst the final schedule is computed over time.

A key observation is that PCP-TDMA has not considered queue lengths. Consequently, it leads to two critical limitations: (i) slot under utilization when scheduled

links have no packets to send, and (ii) network instability, when queues grow to infinity and causes large delay and packet loss. To this end, the next chapter investigates a protocol that considers dynamic queue lengths. In particular, it proposes an algorithm that gives higher transmission priority to links with a higher queue length.

Chapter 5

A Novel Distributed Max Weight Link Scheduler

To date, as discussed in Chapter 2, existing MTR schedulers only aim to minimize the TDMA schedule length. In fact, existing queue aware link schedulers target conventional wireless systems and are impractical. To this end, this chapter studies the aforementioned problem and presents a number of *contributions*. Firstly, for comparison purposes, this chapter outlines a centralized, greedy scheduler called LBC-ALGO. LBC-ALGO is then used to characterize the capacity region of different network topologies. Secondly, this chapter proposes dMaxQ, the first distributed, throughput optimal, link scheduler for MTR WMNs. In fact, dMaxQ is the first distributed MAX-CUT algorithm that incorporates the max weight policy to achieve throughput optimality. In particular, it uses the queue backlog as the link weight and schedules the set of maximum weighted links in each time slot. The performance of dMaxQ is evaluated in both single-hop and multi-hop traffic scenarios.

Compared to similar solutions, e.g., LBC-ALGO, LGS [11], JazzyMAC [26] and ROMA [27], dMaxQ provides the following advantages: 1) each node only communicates with its one-hop neighbors, and it does not require any global topological

information. Consequently, it does not have excessive communication overheads associated with data collection; 2) it uses queue lengths to evenly split nodes into transmitter and receivers at each time slot, which are then paired together to achieve maximum throughput. From simulation results, dMaxQ achieves the same capacity region as the centralized algorithm in all single-hop scenarios, indicating it is a throughput optimal solution.

Next, Section 5.1 describes the MTR network model. After that, the problem is formalized in Section 5.2. Section 5.3 presents a queue aware centralized algorithm, which is used to characterize the stability region of arbitrary MTR WMNs. The proposed distributed policy, i.e., dMaxQ, is outlined in Section 5.4 and analyzed in Section 5.5. Section 5.6 presents the evaluation and results, followed by conclusions in Section 5.7.

5.1 Network Model

Consider a WMN represented by a directed graph $G(V, E)$, where $V = \{1, \ldots, n\}$ is the set of nodes and $E = \{(u, v) \in E \mid u, v \in V\}$ is the set of directed links. Each node has a unique ID. Each node u is equipped with $b_u \geq 1$ directional antennas that share a single channel. Assume $b_u \geq |N(u)|$ for all nodes, where $N(u)$ is the set of neighbors for node u. Each node has a transmission range of r. A link exists if node u and v are within each other's transmission range. Thus, two nodes that are located within a distance smaller than the transmission range are regarded as neighbors. Using the setup presented in [16], each node can achieve concurrent transmission or reception on all its links. That is, at any point in time, a node u can transmit to or receive from $|N(u)|$ neighbors at the same time. However, a node cannot receive *and* transmit simultaneously; this is designated as the *no Mix-Tx-Rx* interference constraint.

Time is divided into slots of equal length, denoted by t. The packets are of unit length. A 0-1 vector with dimension $|E|$, denoted as $A(t)$ is used to represent

packets arrival at each queue/link at the beginning of slot t. The arriving packet is stored in the queue associated with said link. The vector $A(t)$ is calculated based on the arrival rate. Let $Q_{u,v}(t) \geq 0$ represent the number of packets in the queue of link (u, v) at time t. Thus, the queue size vector is denoted by $Q(t) = [Q_{u,v}(t)]$. The vector $D(t) \subseteq \{0, 1\}^{|E|}$ denotes transmitting links at the end of slot t that adhere to the *no Mix-Tx-Rx* constraint. Thus a link (u, v) is active if $D_{u,v}(t) = 1$, for each $D_{u,v}(t) \in D(t)$. Consequently, the dynamic queue length of the network model can be described as $Q(t) = A(t) + Q(t-1) - D(t-1)$.

This chapter considers both single-hop and multi-hop flows. For the single-hop scenario, the Bernoulli process with parameter λ is used as the arrival rate vector. Assume the arrivals are independent and identically distributed. Then, this chapter expands to cases where demands span multiple links. Let F be a set of flows between any pair of nodes. Assume that each node $v \in V$, that has flow $f \in F$, maintains a per-flow queue as q_v^f. In addition, assume that each demand in F is routed over the shortest path. In this multi-hop traffic model, packets are injected at the source node of each flow at a rate of r^f.

To guarantee fairness among flows, the flow rate r^f must maximize $\sum_{f \in F} U^f(r^f)$ [92][99]. Specifically, the goal is to maximize the sum of utility of each source node. The utility function $U^f(r^f) = K \log(r^f)$ of each source node, is strictly concave and continuously differentiable. The constant parameter K allows a link to achieve its full capacity. As per [95], solving $r^f = \arg\max_{r^f \geq 0}(U^f(r^f) - r^f q_s^f)$ yields the following optimal flow rate for a given flow originating at node s: $r^f = K/q_s^f$.

In the multi-hop case, dMaxQ uses queue differential as the nodes' weight. Specifically, the queue differential is defined as the difference between the queue length at one end of a link and the other [25]. Then, to calculate the weight for each node, say i, firstly determine which flow, among all the flows traversing node i, has the maximum queue differential. This flow is denoted as $f^*(i)$; formally, there is $\arg\max_{f \in F}(q_i^f - q_j^f)$, where j represents the next hop neighbor of i on flow f. Then, the weight of node i is set to $W(i) = q_i^{f^*} - q_j^{f^*}$. In other words, the weight of each

node is defined as the maximum differential backlog of its outgoing links.

The following presents standard definitions, see [25][89], used throughout this chapter.

Definition 1. The system is said to be *queue-length stable* if there exist, with probability of one, non-negative real numbers q_j, $j = 1, \ldots, |E|$, such that $\lim_{t\to\infty} \frac{Q(t)}{t} = q_j$. In other words, the network is stable if all queues remain finite.

Definition 2. The *capacity region* (also called the *stability region*) of a given scheduling policy is defined as the set of arrival rate vectors λ under which the network remains queue-length stable.

Definition 3. Let Λ denote the union of the capacity region of all scheduling policies. A scheduling policy is said to be *throughput optimal* if it can achieve the largest possible capacity region Λ.

5.2 Problem Definition

this chapter addresses the following problem. Given queue length information, which refers to the queue length of each link, design a distributed algorithm to derive the throughput optimal scheduling policy for a MTR WMN. Thus, for each time t, the objective of the throughput maximization problem can be expressed as

$$\max \sum_{e\in E} D_e(t)Q_e(t) \tag{5.1}$$

Optimizing Equ. (5.1) is simply the max weight policy and is known to be throughput optimal. However, current approaches are primarily centralized and require instantaneous queue length information, which is not practical in WMNs. In particular, in order to obtain queue information at each time t, a centralized policy will incur many rounds of requests, and signaling overheads as well as propagation and contention delays. These limitations thus motivate the design of a *distributed*

algorithm that is throughput optimal. Thus, the goal is to design an algorithm that approximates the throughput achieved by a centralized policy, which is described next.

5.3 Stability Region of Arbitrary MTR WMNs

First, a centralized scheduling algorithm, called Link Based Centralized Algorithm (LBC-ALGO), is outlined to determine the capacity region of a MTR WMN with an arbitrary topology. LBC-ALGO requires global queue length information, where links are sorted according to their weight or queue length. It then iteratively selects the set of heaviest, non-interfering links for each time slot.

Algorithm 2 specifies LBC-ALGO using C-style pseudo code. Its main aim is to determine the set of transmitting links that satisfies Equ. (5.1) for any time t. To derive the schedule $D(t)$, the algorithm initially defines two empty sets in *Line 4*: $setT(t)$ and $setR(t)$ at the beginning of slot t. The variable $Q'(t)$ in *Line 5* is used to store the queue vector $Q(t)$. The objective is to assign all nodes in V to either $setT$ or $setR$ based on the queue length of each link.

In *Line 6-15*, LBC-ALGO starts by selecting the heaviest link; i.e., one with the longest queue. Once a link is selected, LBC-ALGO sets the link's weight in $Q'(t)$ to zero. Then the second heaviest link becomes the most heaviest link in $Q'(t)$ and will be chosen next. The source and end nodes of a selected link are added into $setT$ and $setR$ respectively if they are not assigned yet. As a result, at *Line 16*, every node is in either $setT$ or $setR$, whereby $setT(t) \cap setR(t) = \emptyset$, and $setT(t) \cup setR(t) = V$. LBC-ALGO then returns the set of transmitting and receiving nodes, i.e., $setT(t)$ and $setR(t)$.

The following propositions provide the capacity upper bound if the network satisfies certain properties.

Proposition 1. The maximum network capacity of a router with at least one incoming and outgoing link is 0.5.

Algorithm 2: LBC-ALGO

Input: $G(V, E)$, queue size vector $Q(t)$
Output: Schedule represented as $setT(t)$, $setR(t)$

```
if |V| ≤ 1 then
    return
else
    setT(t) = setR(t) ← ∅
    Q'(t) ← Q(t)
    while |setT(t)| + |setR(t)| ≠ |V| do
        (u, v) ← LongestQ(Q'(t))
        Q'_{u,v}(t) ← 0
        if u ∉ (setT(t) ∪ setR(t)) then
            setT(t) ← setT(t) ∪ u
            if v ∉ (setT(t) ∪ setR(t)) then
                setR(t) ← setR(t) ∪ v
            end
        end
    end
    return (setT(t), setR(t))
end
```

Proof. Consider a node s that has an incoming and outgoing link. The *no Mix-Tx-Rx* implies that both links must be in a different slot, say S_1 and S_2. At best, all other links can be scheduled in one of these slots, meaning links in S_1, as well as S_2, can only be activated every other slot, which proves the proposition. □

Next, consider the interference set, $I_{s,d}$, in an arbitrary network that contains link (s, d) and every link that interferes with (s, d).

Proposition 2. The maximum throughput for interference set $I_{s,d}$ in any network is 50% if node s and d do not share any common neighbors.

Proof. Let sets X and Y contain the neighboring nodes of s and d, respectively, where $X \cap Y = \emptyset$. Further, without loss of generality, consider an interference set $I_{s,d} = \{(s, d), (x, s), (d, y)\}$, for any one or more node $x \in X$ and $y \in Y$, and thus $x \neq y$. In the best case, two slots are required to activate all links in $I_{s,d}$, i.e., link (s, d) is scheduled in slot S_1, while the remaining links in $I_{s,d}$ in slot S_2. Thus, the throughput is 0.5. □

Proposition 3. The maximum throughput for interference set $I_{s,d}$ in any network is 33% if node s and d do have at least one common neighbor.

Proof. Consider link (s, d) and sets X and Y in the proof of Proposition 2. As node s and d now have common neighbor(s), there exists at least one node $x \in X$ and $y \in Y$ for $x = y$. In other words, the interference set in the proof of Proposition 2 becomes $I_{s,d} = \{(s, d), (x, s), (d, x)\}$. Because link (x, s) and (d, x) cannot be activated in the same slot, three slots are needed to activate link (s, d), (x, s), and (d, x) in $I_{s,d}$. Thus, the throughput is bounded by $\frac{1}{3}$. □

5.4 The Distributed Policy

This section introduces dMaxQ, a distributed policy that emulates LBC-ALGO. Assume each node has a distinct ID. The basic idea is as follows. Initially, all nodes belong to a set called *setU*. Then the nodes in *setU* leaves to join either *setT* or *setR* based on the queue length information collected from their one-hop neighbors. When *setU* is empty, the schedule is complete. All the nodes in *setT* transmit one packet to their neighbors who are in *setR*.

The frame structure used by dMaxQ is shown in Figure 5.1. Each time slot comprises of two sub-slots: *control*, which is used to determine the set of transmitting and receiving nodes, and *data*, which is used for data transmissions. Each control sub-slot is further divided into a number of mini-slots that are used to transmit signaling messages in the following three stages of dMaxQ, namely, (i) Stage-1: **initialize**, (ii) Stage-2: **notify**, and (iii) Stage-3: **compare**. Stage-1 requires M mini-slots and is used by nodes to exchange queue information and for certain eligible nodes to enter *setT*. On the other hand, Stage-2 takes two mini-slots and is for nodes that have joined *setT* or *setR* to advise their neighbors which set they have entered. Finally, Stage-3 incurs one mini-slot, and is used by the remaining nodes who are neighbors of the nodes that just entered *setT* to determine whether they are to join *setT* or *setR*.

As will become clear later, Stage-2 and Stage-3 operate alternately for i times until $setU$ is empty, where i is defined as the iteration number; explained in detail later. The control messages used by dMaxQ are listed on Table 5.1.

As mentioned, the mini-slots in Stage-1 are used to exchange queue information; i.e., nodes transmit their queue information to their neighbors in a collision-free manner. A key question here is how nodes are assigned a slot in Stage-1 and indeed, how many slots are required to ensure all nodes are given a slot. To this end, in order to derive the required schedule, ALGO-1 [23], a heuristic solution for the well-known, NP-complete MAXCUT problem is used. Briefly, in each slot, it determines the maximal set of transmitting nodes that adhere to the no Mix-Tx-Rx constraint or equivalently, the MAXCUT whereby nodes are separated into two sets, say Set1 and Set2, and nodes in Set1 transmit to those in Set2. It stops once all links are assigned a slot. When a node is assigned a slot, it transmits on all its out-going links. The length of the derived schedule or superframe length, in number of slots, is denoted by M. Both the theoretical and practical results in [23] show that the length of the schedule derived by ALGO-1 is related to the node density and the degree of nodes. Assume ALGO-1 is run at network deployment time or whenever the topology changes, which is likely to be infrequent as nodes are static.

In Stage-1, explained later, a node, say s, is required to determine its own weight $W(s)$ and inform all its one-hop neighbors. Here, the weight is computed as $W(s) = MAX(Q_{s,N(s)})$, where $Q_{s,N(s)}$ is a set containing the queue length for each link to neighbors in $N(s)$. In words, $W(s)$ is set to the maximum queue length at node s. As an example, if node 1 has three outgoing links with queue lengths of 1, 4 and 5 with corresponding neighbor 2, 4 and 3, respectively, as shown in Figure 5.2(a), then $W(1) = 5$, and this weight will be transmitted to node 2, 4 and 3. With a slight abuse of notation, $W(s)$ is also used as a message transmitted by node s to its neighbors.

There are three key stages of dMaxQ: **initialize**, **notify** and **compare**. The goal of the **initialize** stage is to select the first batch of nodes to join $setT$; see

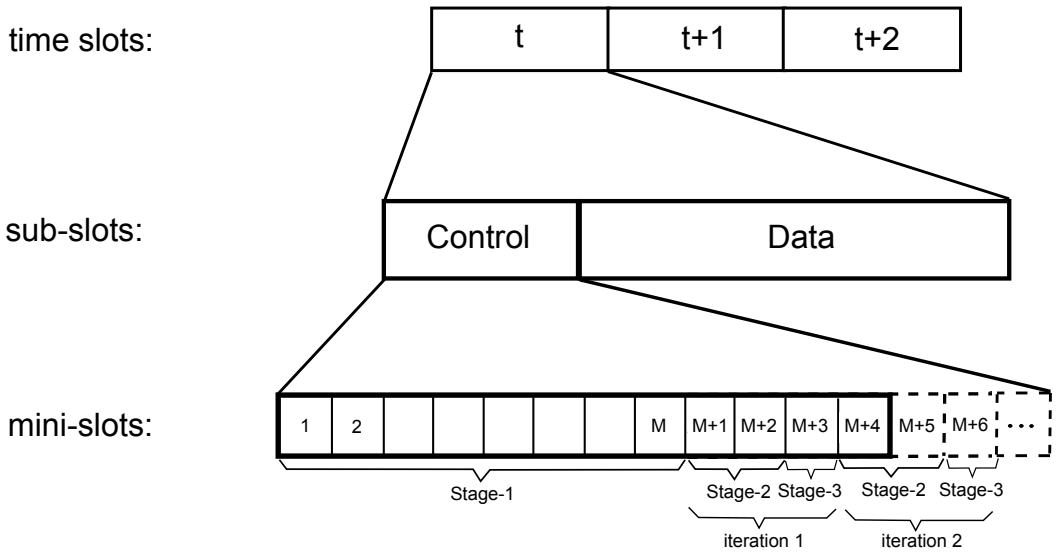


Figure 5.1: Time slot structure

Table 5.1: Messages used in dMaxQ

Message	**Description**
$W(s)$	The weight of node s, which is equal to the maximum queue length among all outgoing links of s.
$M\{INsetT\}$	This message notifies the receiver that the sender has entered $setT$.
$M\{INsetT, ENTERsetR\}$	This message notifies the receiver that the sender has entered $setT$ and also command the receiver to join $setR$.
$M\{INsetR\}$	It is used to notify the receiver that the sender has entered $setR$.
$OUTq\{Q_{s,d}\}$	This message is sent by a node s that just entered $setT$ to its neighbor d in $setU$. The message indicates the number of packets awaiting transmission from node s to d.

Algorithm 3. Recall that in the first M mini-slots, nodes transmit their weight to their neighbors. Once M mini-slots complete, every node, say s, knows its own weight $W(s)$ and the set of weights $\{W(u) \mid u \in N(s)\}$ of its one-hop neighbors $N(s)$. This information is then used by nodes to place themselves in $setT$ by comparing their weight to that of their neighbors based on the rules in Algorithm 3. If a node, say s, finds itself to have the maximum weight among all its neighbors, then node s enters $setT$.

Algorithm 3: Stage-1: initialize

Input: node s, neighbors $N(s)$, $W(s)$ and $W(d), \forall d \in N(s)$
Output: $set(s)$

```
1  x ← arg max_{d∈N(s)} W(d)
2  if W(s) > W(x) then
3  |   s departs from setU for setT
4  |   set(s) ← "setT"
5  else if W(s) = W(x) and s > x then
6  |   s departs from setU for setT
7  |   set(s) ← "setT"
8  else
9  |   set(s) ← "setU"
10 end
11 return set(s)
```

In *Line 1* of Algorithm 3, the neighbor node x that has the maximum weight is found. Then in *Line 2-4*, if a node finds its own weight $W(s)$ to be greater than that of node x, it places itself in $setT$. In *Line 5-8*, dMaxQ also considers the case where there is a tie. In this case, the node with the higher ID enters $setT$. Otherwise, node s stays in the set $setU$; see *Line 9*.

Figure 5.2(a) shows an example to explain this phase of dMaxQ. The number next to each link represents its queue length. Observe the $W(.)$ of each node. Figure 5.2(b) shows the transmission of the $W(.)$ message between neighboring nodes. After obtaining the required weight, each node compares its own weight with that of its neighbors. As a result, node 1 and 5 enter $setT$ because these two nodes have the maximum weight among their neighbors. In the example, a gray circle represents a

node in $setU$, a black circle is a node in $setT$.

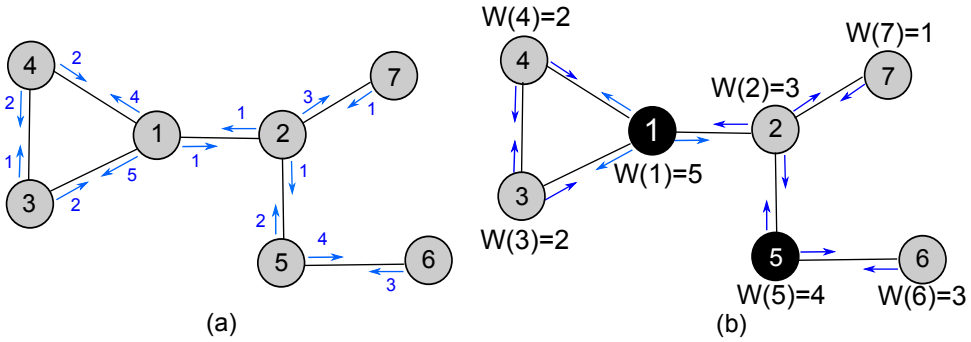


Figure 5.2: An example of dMaxQ in Stage-1

Next is the **notify** stage. A node that has entered $setT$, say s, transmits two kinds of messages: (i) $M\{INsetT, ENTERsetR\}$ is sent to neighbor x of link (s, x) with the longest queue length; and (ii) $M\{INsetT\}$ is transmitted to all neighbors, except node x, to inform them it has joined $setT$. A node in $setU$ that receives the command "ENTERsetR" enters $setR$ and then transmits $M\{INsetR\}$ message to all its neighbors to confirm the move.

Note that this stage contains two mini-slots: $[M + 1, M + 2]$. The reason is because the messages sent by nodes in $setT$ and $setR$ cannot be transmitted in the same mini-slot. If two neighboring nodes, say a and b, are in $setT$ and $setR$ respectively, then the messages sent by node a and b will collide because it violates the *no Mix-Tx-Rx* constraint. Hence, nodes in $setT$ and $setR$ are required to transmit in mini-slot $[M + 1]$ and $[M + 2]$, respectively.

The **notify** stage is formally defined in Algorithm 4. For a given node s that is in $setT$, see *Line 1-10*, node s firstly finds the neighbor node x with the largest $Q_{s,x}$ value. Node s then sends the message $M\{INsetT, ENTERsetR\}$ to node x. This message notifies node x that node s will be in $setT$ and also instructs node x to enter $setR$. For other neighbors, i.e., $N(s) - x$, node s transmits a $M\{INsetT\}$ message. Then in *Line 11-13*, if node s received a $M\{INsetT, ENTERsetR\}$ message and also belongs to $setU$, it will leave $setU$ to join $setR$; as instructed by the "ENTERsetR" command. Nodes that have joined $setR$ then send the message $M\{INsetR\}$ to all their neighbors; see *Line 14*.

Algorithm 4: Stage-2: notify

Input: node s, $set(s)$, $N(s)$, $Q_{s,d}, \forall d \in N(s)$
Output: $set(s)$

if $set(s) =$ *"setT"* **then**
  $x \leftarrow \arg\max_{d \in N(s)} Q_{s,d}$
  for $d \in N(s)$ **do**
    if $d = x$ **then**
      Transmit $M\{INsetT, ENTERsetR\}$ to d
    else if $d \neq x$ **then**
      Transmit $M\{INsetT\}$ to d
    end
  end
end
if s *receives* $M\{INsetT, ENTERsetR\}$ ***and*** $set(s) =$ *"setU"* **then**
  s departs from *setU* for *setR*
  $set(s) \leftarrow$ *"setR"*
  Transmit $M\{INsetR\}$ to all $d \in N(s)$
end
return $set(s)$

Figure 5.3 illustrates the **notify** stage. As shown in Figure 5.3(a), in mini-slot $M+1$, nodes 1 and 5 send the message $M\{INsetT, ENTERsetR\}$ to node 3 and 6 as they are the end point of the link with the longest queue, and the message $M\{INsetT\}$ is sent to node 2 and 4. In mini-slot $M+2$, see Figure 5.3 (b), node 3 and 6 then send the $M\{INsetR\}$ message to their neighbors that they have joined *setR*.

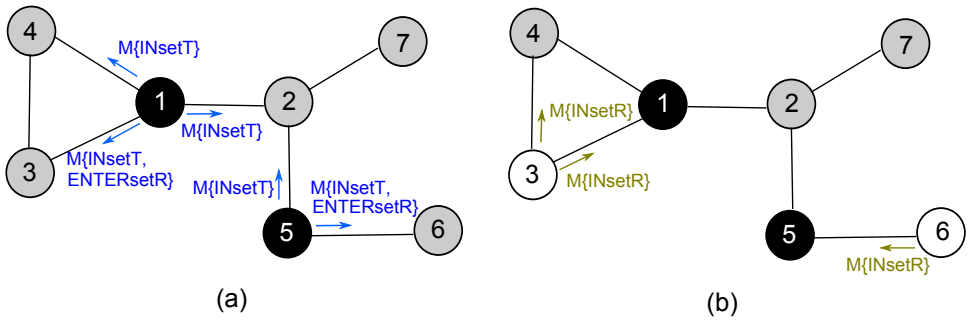


Figure 5.3: An example of dMaxQ operating in Stage-2 (iteration 1)

Finally, Stage-3 or **compare** stage takes place in mini-slot $[M+3]$. Note, this stage only incurs one mini-slot. At this point nodes have learned whether their neighbors are in *setT*, *setR* or *setU*. This stage aims to resolve nodes belong to *setU*

that are neighbors of nodes in $setT$. Referring to Figure 5.3(b), this means only nodes 1 and 5, both are in $setT$, and nodes 2 and 4, who are in $setU$, will participate in Stage-3. Node-7 will be handled in subsequent iterations.

To explain Stage-3, let us consider a node s in $setU$ that is a neighbor of nodes in $setT$. It needs to determine which set, i.e., $setT$ or $setR$, it should join. To make this decision, node s compares the values in the following sets: $T(s)$ and $R(s)$. Specifically, the set $T(s)$ contains the outgoing queue lengths of node s, and the set $R(s)$ records the incoming queue lengths reported in the $OUTq$ message sent by node s's neighbors that are in $setT$. In particular, $OUTq$ messages enable node s to determine how many packets it will receive should it move to the set $setR$.

An $OUTq$ message is defined as "$OUTq\{Q_{x,s}\}$", where x and s represent the source and destination nodes of link (x, s), and $Q_{x,s}$ is the queue length of the link from x to s. Here, x is a neighbor of node s that is in $setT$. Upon receiving an $OUTq$ message, node s stores its content in the set $R(s)$; as we will see later this thus allows node s to have a record of the number of packets awaiting to be delivered to node s.

Initially, there is $T(s) = Q_{s,N(s)}$, meaning $T(s)$ is set to the queue length of each outgoing link to each neighbor. After that, whenever a neighbor enters $setT$, node s removes it from $T(s)$. This is because node s is no longer permitted to transmit to this neighbor due to the no Mix-Tx-Rx constraint. Given $T(s)$ and $R(s)$, node s compares the biggest element in both sets. A greater value in $T(s)$ indicates that it has a longer outgoing queue and thus enters $setT$. Otherwise, it moves into $setR$.

As shown in *Line 1-4* of Algorithm 5, if node s is in $setT$, it transmits an $OUTq$ message containing its outgoing queue length to each neighbor d that is in $setU$. Otherwise, from *Line 5-6*, if node s is in $setU$, and its neighbor d is in $setT$, node s would have received a $M\{INsetT\}$ message earlier from its neighbor d in Stage-2. Node s then updates its set $T(s)$ by deleting the queue length $Q_{s,d}$; see *Line 7*. This means node s is not allowed to transmit to neighbor d. As node d is in $setT$, it would have sent an $OUTq$ message to node s in its *Line 3*. Upon receiving an $OUTq$

message, node s adds $Q_{d,s}$ to the set $R(s)$; see *Line 8*. Finally, dMaxQ compares $MAX(T(s))$ and $MAX(R(s))$; *Line 11-17*. Node s enters $setT$ if set $T(s)$ holds an element with a higher value. Otherwise, it becomes a receiver.

Algorithm 5: Stage-3: compare

Input: node s, $set(s)$, neighbors $N(s)$, and the set of queue lengths $Q_{s,N(s)}$
Output: $set(s)$

```
1  if set(s) = "setT" then
2  |   for d ∈ N(s) ∩ setU do
3  |   |   s transmits OUTq{Q_{s,d}} to d
4  |   end
5  else if set(s) = "setU" then
6  |   for d ∈ N(s) ∩ setT do
7  |   |   T(s) ← Q_{s,N(s)} \ Q_{s,d}
8  |   |   R(s) ← R(s) ∪ Q_{d,s}
9  |   end
10 end
11 if MAX(T(s)) > MAX(R(s)) then
12 |   s leaves setU for setT
13 |   set(s) ← "setT"
14 else
15 |   s leaves setU for setR
16 |   set(s) ← "setR"
17 end
18 return set(s)
```

Figure 5.4 shows the **compare** stage for the foregone example. We can see in Figure 5.4(a), in mini-slot $M + 3$, node 1 sends an $OUTq$ message to node 2 and 4; similarly, node 5 also sends an $OUTq$ message to node 2. Observe that all the receivers of the $OUTq$ message are in $setU$. Conversely, only nodes in the set $setT$ transmit to neighbors in $setU$. In other words, only one mini-slot is required. From Figure 5.4(b), we see that the value(s) of $T(2)$, $R(2)$ and $T(4)$, $R(4)$ respectively. Based on $T(.)$ and $R(.)$ value, node 2 and node 4 enter $setT$ and $setR$, respectively, because $MAX(T(2)) = 3$ is greater than $MAX(R(2)) = 2$ while $MAX(R(4)) = 4$ is greater than $MAX(T(4)) = 2$. Observe that node 2 is in $setT$ and its neighbor node 7 is in $setU$. This means Stage-2 and Stage-3 will have to be repeated to determine node 7's membership; i.e., $setT$ or $setR$.

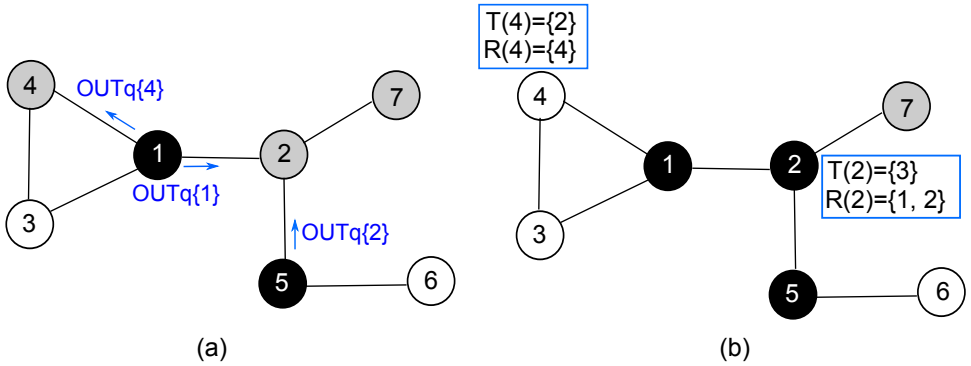


Figure 5.4: An example of dMaxQ in Stage-3 (iteration 1)

At this point, a subset of nodes have joined $setT$ in Stage-1 (node 1 and 5 in the earlier example), and all of their one-hop neighbors have entered either $setT$ or $setR$ in Stage-2 and 3 (node 2, 3, 4 and 6), while other nodes are still in $setU$ (node 7). From this point onwards, Stage-2 and Stage-3 operate alternately for the remaining nodes in $setU$ for i times. The iteration number i is set to be equal to the network diameter. However, in practice, the iteration number can be set to a significantly smaller value than the network diameter. For example, from the extensive simulations outlined in Section 5.6, the average number of iterations required for every node to join $setT$ or $setR$ is three, while the network diameter is about 10.

In the previous example, $i = 4$ because the longest shortest path has a length of four hops, e.g., between node 3 and 6 and node 4 and node 6. Continuing the example, in Figure 5.5(a), as node 2 has entered $setT$, in mini-slot $M + 4$, node 2 sends $M\{INsetT\}$ to node 1 and 5, and $M\{INsetT, ENTERsetR\}$ to node 7. Then in Figure 5.5(b), node 4 and 7 send $M\{INsetR\}$ message to their neighbors in mini-slot $M + 5$ because they have joined $setR$; a white circle denotes a node in $setR$. Continuing the example, after Stage-2 iteration 2 is finished, Stage-3 iteration 2 operates in mini-slot $M + 6$. However, as each node knows that none of its neighbors are in $setU$, it does not send an $OUTq$ message. This is true for the remaining iterations, i.e., $[M + 7, M + 12]$.

At the end of the control sub-slot, data transmissions begin. In other words, all the nodes in $setT$ transmit one packet to each of their neighbors that are in $setR$.

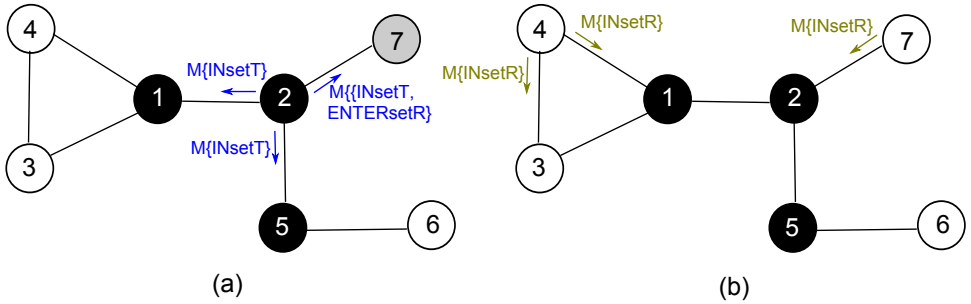


Figure 5.5: Continuation of dMaxQ in Stage-2 (iteration 2)

Figure 5.6 shows how packets are transmitted for the earlier example.

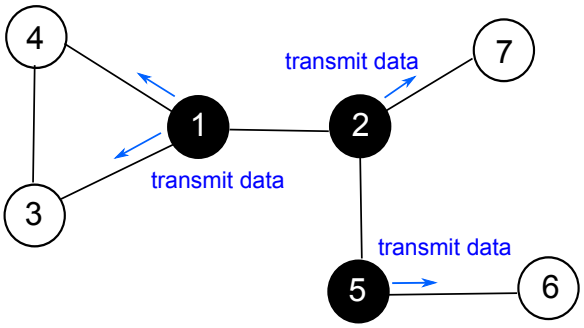


Figure 5.6: Transmissions in the data slot

To sum up, Figure 5.7 provides a state diagram to illustrate how dMaxQ is used to produce a schedule. Initially, all nodes belong to $setU$. In the first stage, each node exchanges its weight with its neighbors and determines whether it has the maximum weight as compared with its neighbors. If a node has the heaviest weight, then it joins $setT$ directly. At the next stage, the node that entered $setT$ transmits $M\{INsetT\}$ and $M\{INsetT, ENTERsetT\}$ to its neighbors. Once a node in $setU$ receives the message $M\{INsetT, ENTERsetR\}$, it enters $setR$ and transmits $M\{INsetR\}$ to inform its neighbors. For the nodes that are still in $setU$, they will go through the third stage. At this final stage, the nodes in $setT$ send an $OUTq$ message to their neighbors that are in $setU$. If a node in $setU$ receives an $OUTq$ message, the content of $OUTq$ will be stored in the set R. After comparing $MAX(T)$ with $MAX(R)$, the decision to join $setR$ or $setT$ can be made. However, if a node does not receive any $OUTq$ messages from its neighbors, the node goes back to Stage-2.

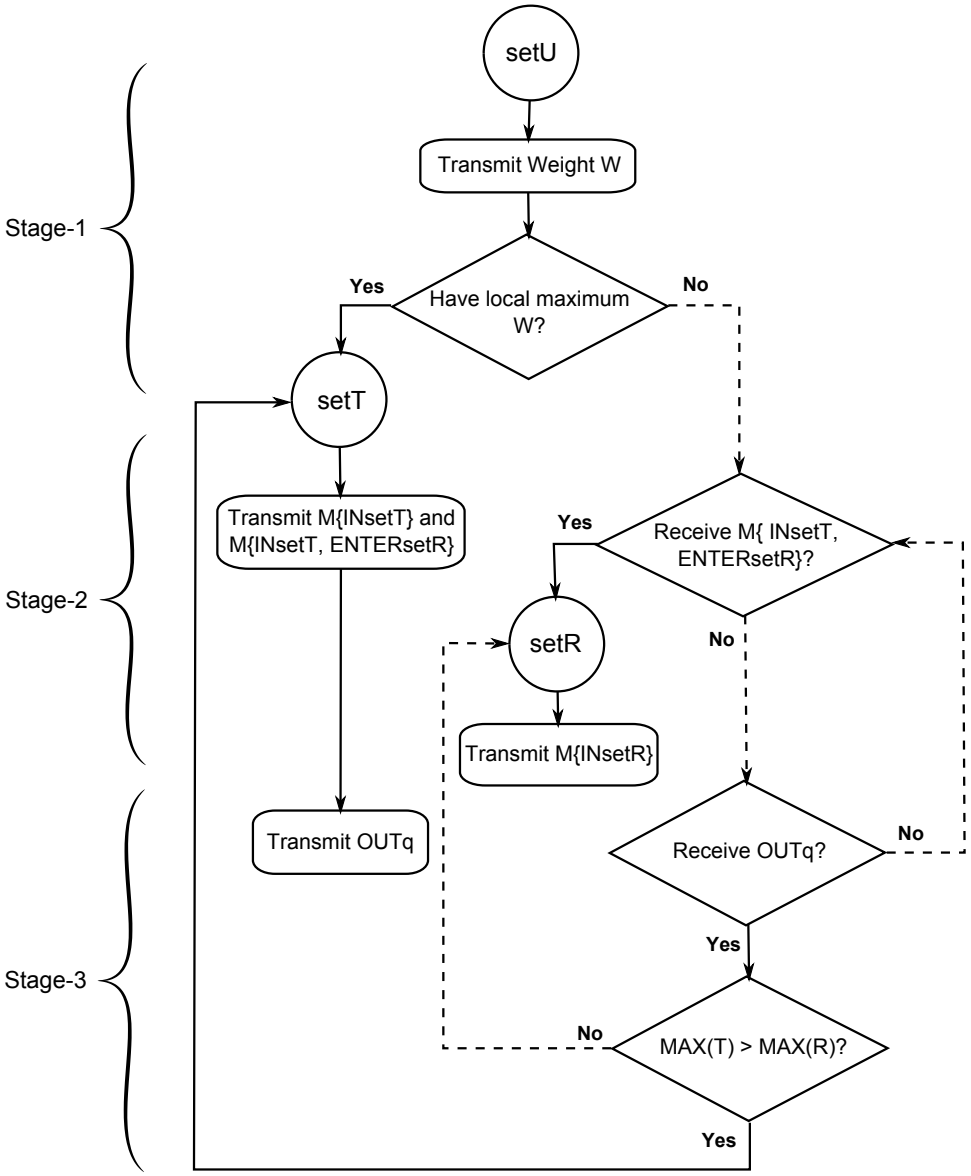


Figure 5.7: State Diagram of dMaxQ

5.5 Analysis

This section analyzes dMaxQ from the following aspects: the correctness, the number of signaling messages generated, the bound on iteration number i, and selecting the heaviest link using only local information. These properties of dMaxQ are listed below.

Theorem 1. dMaxQ and its produced schedule conform to the *no Mix-Tx-Rx constraint.*

Proof. To show that dMaxQ conforms to the constraint, consider each stage of dMaxQ: In Stage-1 (**initialize**), it uses the schedule derived by ALGO-1 [23] to exchange queue information. As proven in [23], ALGO-1 derives a schedule that is collision free. Based on the rules stated in Stage-1, we know that for any node a that joins $setT$ in Stage-1, none of a's one-hop neighbors can join the same set in this stage. That is to say, if node a is in $setT$, none of their respective neighboring nodes are in $setT$. In Stage-2 (**notify**), as dMaxQ requires node a in $setT$ and b in $setR$ to transmit $M\{INsetT\}$(or$M\{INsetT, ENTERsetR\}$) and $M\{INsetR\}$ in two different mini-slots, a and b will not transmit and receive the said notification messages in the same mini-slots even if a and b are neighboring nodes, which avoided the *Mix-Tx-Rx* interference. In Stage-3 (**compare**), there are three sets of nodes: $setT$, $setR$ and $setU$. Only nodes in the first set are in the *transmitting* state, and they only communicate with nodes in the last set that is in the *receiving* state. Thus, all nodes conform to the said constraint, proving Stage-3 is collision free.

Notice that each node knows its own state, i.e., "setT" or "setR", and its neighbors' state. Therefore, each node in $setT$ will only transmit to all neighbors in $setR$, and thus the produced schedule conforms to the *no Mix-Tx-Rx constraint.* □

Next, the focus is on the number of signaling messages issued by each node.

Proposition 4. dMaxQ requires each node a to transmit in total no more than $3|N(a)|$ signaling messages.

Proof. In the **initialize** stage, $|N(a)|$ messages are transmitted because each of node a's outgoing links is activated to transmit its weight $W(a)$ to each one-hop neighbor. The **notify** stage requires another $|N(a)|$ messages as a node transmit a $M\{INsetT\}(M\{INsetT, ENTERsetR\})$ or a $M\{INsetR\}$ to every neighbor. For the **compare** stage, it requires at most $|N(a)|$ messages since a node transmits an $OUTq$ message to each of its neighbors that belongs to $setU$. Note, each node needs to transmit the above messages in at most one iteration. Thus, the total number of signaling messages required for each node is at most $3|N(a)|$. □

The following definitions will be required to prove the bounds on the number of iterations. Denote the first running of Stage-2 and Stage-3 as 'Stage-2 (Stage-3) iteration 1'. Let G_0 be the set of nodes that have entered $setT$ in Stage-1, and G_i as the set of nodes that join either $setT$ or $setR$ in Stage-2 and Stage-3 of iteration i. Note that all the nodes in G_i are the one-hop neighbors of nodes in G_{i-1}.

Proposition 5. The number of iterations, denoted as i, required by dMaxQ to schedule every link in a network with $|V|$ nodes is bounded by $1 \leq i \leq |V| - 1$.

Proof. For any given network, we know that during each iteration of Stage-3, the nodes in G_z will transmit $OUTq$ messages to its one-hop neighbors in $setU$. This allows nodes in $setU$ to decide whether they should join $setT$ or $setR$. If every node in the network has at least a one-hop neighbor in G_0, only one iteration is required to schedule all links. For example, if node 7 of Figure 5.2 is not present, dMaxQ requires one iteration. Otherwise, i is bounded by the minimum hop between node s and d, where $s \in G_0$ and $d \notin G_0$. Consider a chain topology. The maximum number of iterations occurs when the first node of the chain has the highest weight followed by its next hop neighbor and so forth. In this case, only the first node is in G_0. Since the furthest node at the other end of the chain is $|V| - 1$ hops away, it requires $|V| - 1$ iterations for every node of the chain to enter $setT$ or $setR$. In other words, stage 2 and 3 of dMaxQ will proceed from the first and heaviest node to the other end of the chain whereby in each iteration only one node leaves $setU$. Thus,

the number of iterations is bounded by $1 \leq i \leq |V| - 1$. □

Using Proposition 5, we can calculate the upper bound of the number of mini-slots needed for dMaxQ as $M + [(2+1) \times (|V| - 1)]$, where M is required in Stage-1, two and one slots are needed by Stage-2 and Stage-3 respectively. The maximum iteration i is $|V| - 1$.

Lemma 1. dMaxQ selects the heaviest weighted link for transmission using only 1-hop neighbors' weight.

Proof. Consider a node a. By assumption, knowing all its 1-hop neighbors' weight means it knows all the longest queue lengths of each of its neighbors' out-going links. In Stage-1, for node a, if any neighbor's weight $W(N(a))$ is more than $W(a)$, then node a will not transmit. Conversely, if $W(a)$ is the highest amongst its 1-hop neighbors, then node a's neighbors will be aware of $W(a)$, and therefore, node a will be a transmitter and in Stage-2, the corresponding end of the link with weight $W(a)$, say node in $setU$, will become a receiver. Consequently, amongst nodes in $N(a) \cup a$, only the one with the heaviest link will initiate transmission. □

Note that if a neighbor of nodes in $N(a)$ that is two hops away from node a, say b, has the heaviest link, then it will transmit. This does not affect node a's transmission as node in $setU$ will be in receive mode. Moreover, it will not prevent node b from transmitting. In case of a tie, then the node with the highest ID wins; recall that a node knows its neighbors' ID.

Proposition 6. For any N hops MTR WMNs, the heaviest link will be activated in each hop.

Proof. This proposition is prove by induction. For $N = 1$, this proposition is true because all nodes are aware of each other's weight, and hence, only the node with the highest weight will become a transmitter. Let $N \leq l$ be an integer, where $l \geq 2$. the aim is to prove that the proposition is true for $N = l + 1$. Consider node v_l located at the l-th hop. By the induction hypothesis, node v_l can either be a transmitter

(Case-1) or a receiver (Case-2). In other words, for Case-1, node v_l has the highest $W(v_l)$ value, whilst in the latter, a neighbor located at the v_{l-1}-th or v_{l+1}-th hop has the highest weighted link. This proposition is true for both cases. First, consider Case-1. Node v_l is a transmitter to a node in v_{l-1} or v_{l+1}, which proves that the heaviest link is activated in hop l and $l+1$. In Case-2, using Lemma 1, node v_{l+1} knows the weight of node v_l and v_{l+2}. Similarly, node v_l knows the weight of node v_{l+1} and v_{l-1}. If node v_l has the highest $W(v_l)$, then node v_l is a transmitter. Otherwise, either v_{l-1} or v_{l+1} is a transmitter or both; in all cases, node v_l is the receiver of the heaviest link activated at hop $l-1$ and/or $l+1$. Consequently, the heaviest link is activated at each hop for any N. □

5.6 Evaluation

In this section, to investigate the performance of dMaxQ via simulation, MatGraph [97], a Matlab toolkit that works with simple graphs is used. In all topologies, the nodes are fixed. the experiments compare dMaxQ against LBC-ALGO, node based centralized algorithm (NBC-ALGO), JazzyMAC [26] and ROMA [27]. The basic idea of NBC-ALGO is similar to that of LBC-ALGO, which is introduced in Section 5.3. The difference is that instead of using the maximum queue length amongst all flows, NBC-ALGO sums up all queue lengths. For JazzyMAC, assume max_slot is set to 20.

In the sequel, both one-hop and multi-hop traffic cases are investigated. The former allows us to study the performance of dMaxQ in scenarios that are independent of any routing or transport protocols. Consequently, it only focuses on dMaxQ's ability to keep queue stable and also maximize throughput. This is in contrast to the multi-hop case where transport layer behaviors have a non-negligible impact on the link schedule.

5.6.1 Single-hop Traffic

First, consider the one-hop traffic model where all routes consist of one link. In this respect, the transmitting end of a link is denoted as its source, and the corresponding end as the receiver. A packet arrival occurs with probability λ at the beginning of a time slot. The arrivals are independent and identically distributed according to a Bernoulli process. Consider two different topologies: bipartite and random. The bipartite topology is a grid consisting of 4×4 nodes. Each node communicates with its neighbors located at its "left", "right", "top" and "bottom". The random topology is generated by randomly placing 50 nodes in a 100m $\times$ 100m square area, and the transmission range is set to $r = 30m$. The following figures plot the mean total queue backlog summed over all links and the average delay over all transmitted packets as the arrival rate λ increases. The error bars shown in the line graphs indicate the 95% confidence interval. When λ approaches a certain limit, the average total backlog as well as the delay will increase to infinity. This limit can then be viewed as the boundary of the capacity region. The results are collected after 100000 time slots.

Figure 5.8 and 5.9 show the performance of each algorithm in terms of queue length and delay. We can see LBC-ALGO, NBC-ALGO and dMaxQ achieve the largest throughput of almost 50% in the grid topology. According to Proposition 2, this percentage, i.e., 50%, is the maximum achievable throughput for this topology.

In addition, we can see that JazzyMAC has the lowest performance amongst all five algorithms. This is because JazzyMAC requires a node to collect all tokens for its links before transmission. This design feature is especially limiting if an outgoing link can start transmission without violating the no Mix-Tx-Rx constraint but is blocked by adjacent links that have yet to receive their token. Consequently, a number of time slots are wasted. Although ROMA does not require any signaling, it does not take the exact queue length of each link into account but relies only on coarse weights. In the experiments, ROMA can achieve only approximately 60% of

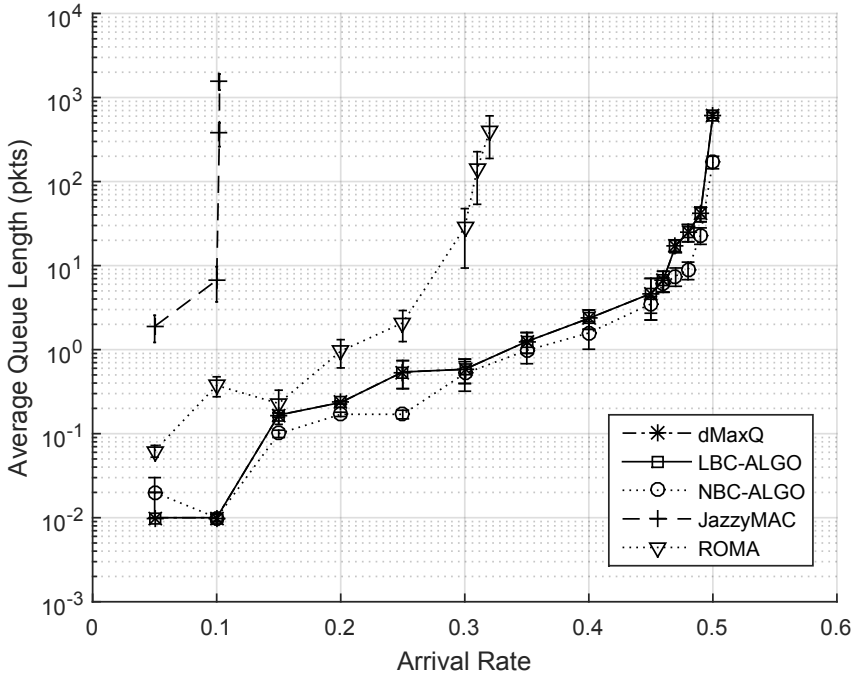


Figure 5.8: Average queue under increasing arrival rate for a grid topology

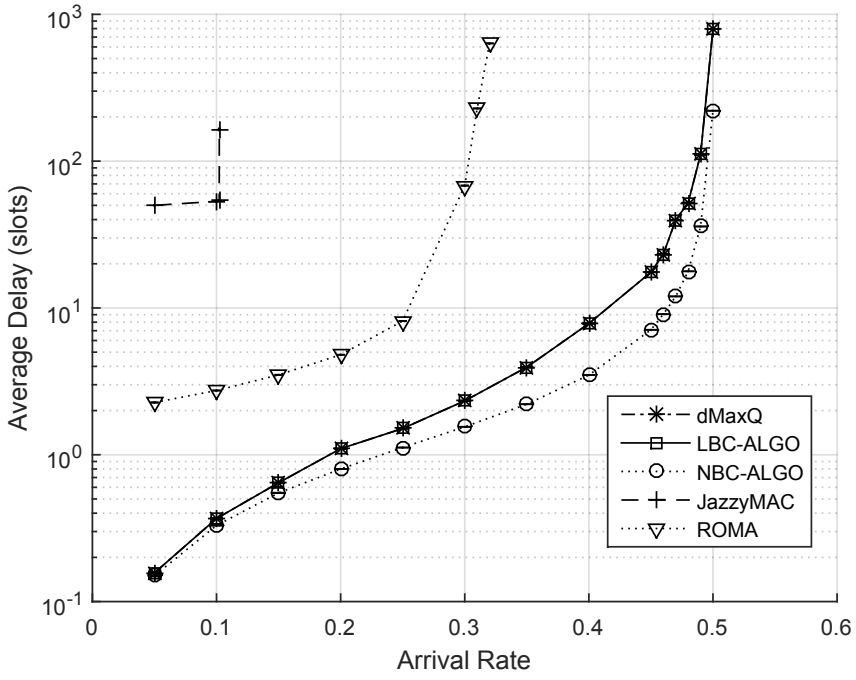


Figure 5.9: Average delay under increasing arrival rate for a grid topology

the throughput as compared to dMaxQ, LBC-ALGO and NBC-ALGO. As shown in Figure 5.9, dMaxQ and LBC-ALGO produce the same delay, slightly worse than NBC-ALGO, which produces the lowest delay. This is because in single-hop grid topologies, LBC-ALGO activates the set of links with the heaviest weight, while NBC-ALGO aims to activate the maximum number of links, which means that the maximum number of packets are served each slot When compared to JazzyMAC and ROMA, nodes using dMaxQ have much lower delays.

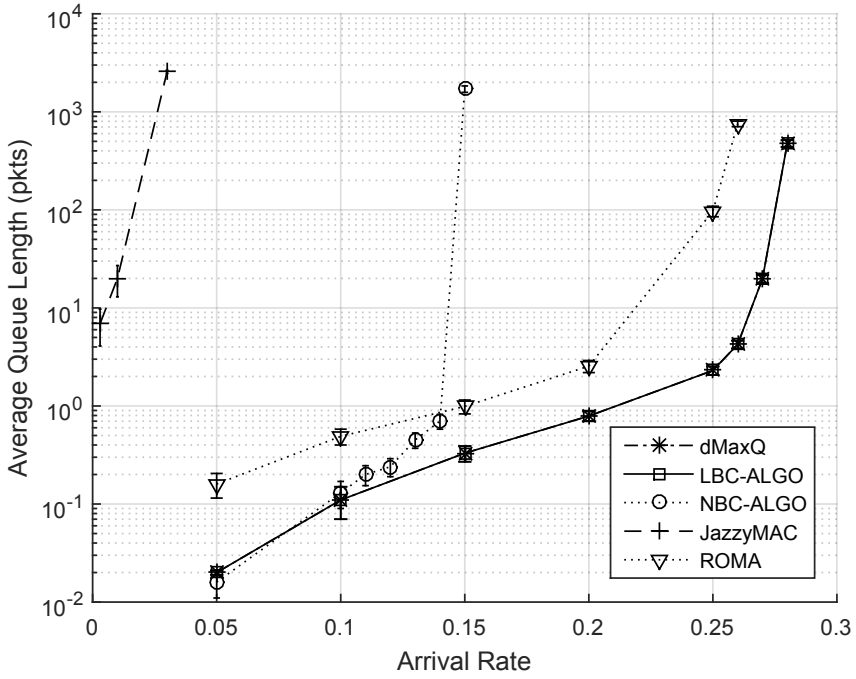


Figure 5.10: Average queue under increasing arrival rate for a random topology

Figure 5.10 and 5.11 present the results for random topologies. Notice that the performance of all algorithms reduces as compared to the bipartite case, especially JazzyMAC and NBC-ALGO. For JazzyMAC, its capacity region decreased by 70%. This is because the random topology has a maximum node degree of 30, which is much higher than 4 for the grid topology. As a result, nodes using JazzyMAC spend a significant amount of time collecting tokens from all their neighbors. Recall that the definition of weight in NBC-ALGO is the sum of the queue length of every outgoing link. This means NBC-ALGO may not schedule the node with maximum per-link queue into a transmitting set. Thus, NBC-ALGO is not using the max

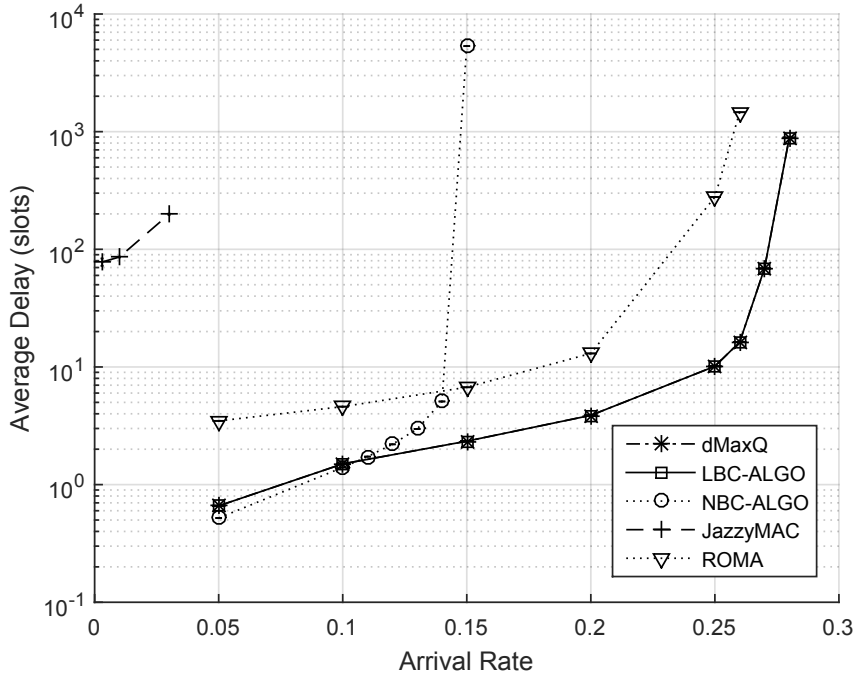


Figure 5.11: Average delay under increasing arrival rate for a random topology

weight policy. Consequently, it only achieves half the throughput of LBC-ALGO.

Table 5.2: Capacity region results

Topology	dMaxQ	LBC-ALGO	NBC-ALGO	JazzyMAC	ROMA
Grid	0.49	0.49	0.5	0.1	0.31
Random	0.28	0.28	0.15	0.03	0.25

From the results of grid and random topology, we can see that dMaxQ ensures all queues are stable with the same traffic load as the centralized algorithm in both scenarios. According to Table 5.2, dMaxQ achieves the same capacity region as LBC-ALGO. This result indicates that the performance of dMaxQ is approximately similar to the centralized approach. In other words, dMaxQ is throughput optimal. In term of delay, Figure 5.11 shows that dMaxQ and LBC-ALGO produce the same lowest delay, outperforming other algorithms for arrival rates larger than 0.15.

5.6.2 Multi-Hop Traffic

In experiments concerning multi-hop traffic, 50 nodes are randomly placed on a square area of $100 \times 100m^2$, with transmission range $r = 30m$. Source and destina-

tion node pairs are randomly selected with the number of flows ranging from 5 to 50. this section first plots the average queue and delay at the end of the 100000-th time slot. Then it compares the average per-source throughput of each algorithm. Note, LBC-ALGO and NBC-ALGO are modified to use the queue differential instead of the queue length.

Lastly, the throughput is only measured at the source. This is because the flow rate is determined by the queue length at the source, which in turn is determined by how often the algorithm empties the queue. At every stage, dMaxQ aims to schedule the links with the largest queue differential. This means the faster dMaxQ empties queues, upstream nodes will be able to inject packets more frequently, and thus improve throughput and guarantees queue stability.

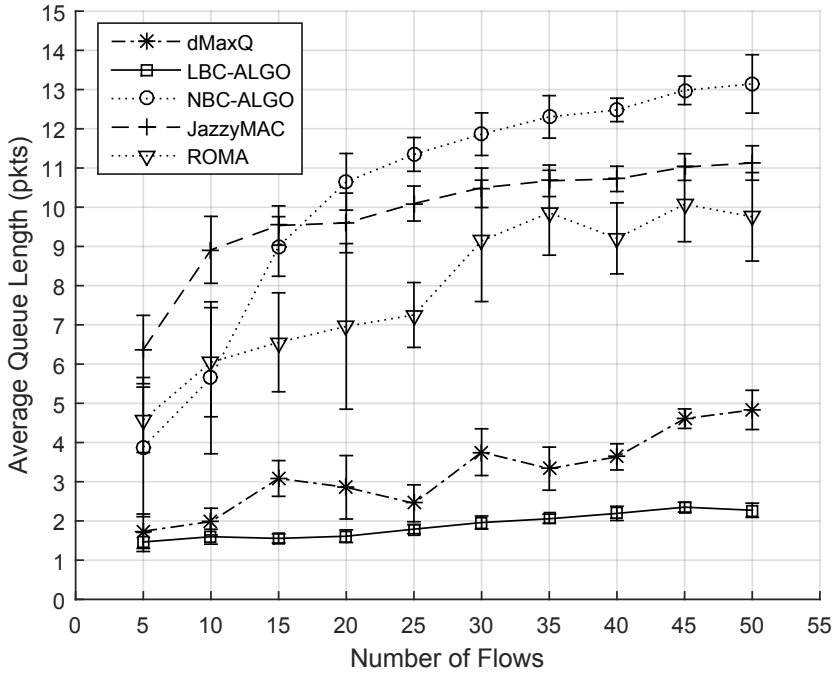


Figure 5.12: Average queue length under increasing number of flows for a random topology

Figure 5.12 and 5.13 show the average per-link queue length and per-packet delay of each algorithm for experiments over random topologies. We can see that the queue lengths of dMaxQ and LBC-ALGO are similar, with values ranging between 1.5 to 4.9. On the other hand, JazzyMAC, ROMA and NBC-ALGO show higher queue

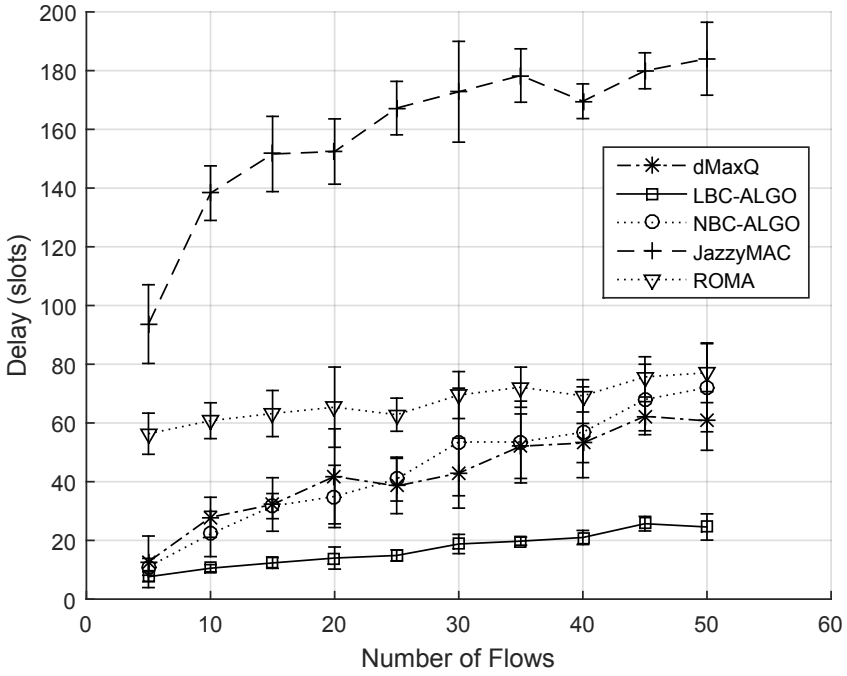


Figure 5.13: Average delay under increasing number of flows for a random topology

lengths, ranging from 4 to 13. We can see that the queue length of all algorithms is comparatively steady as the number of flows increases from 20 to 50. This agrees with the theoretical result of [25]. Also, note that the source controls its rate using the function $r^f = K/q_s^f$. Thus, when the queue corresponding to a flow builds up, the flow's arrival rate decreases and vice-versa. Consequently, the function always ensures queues are stable; i.e., it prevents congestion from happening.

In term of delay, Figure 5.13 shows that dMaxQ and LBC-ALGO yield similar performance, while NBC-ALGO generates the lowest delay. Note that dMaxQ outperforms the other two distributed algorithms, i.e., JazzyMAC and ROMA.

Figure 5.14 plots the mean throughput and the 95% confidence intervals. This throughput represents the amount of data injected into the network and is defined as $\sum_{f=1}^{|F|} r^f$. Among the three distributed approaches, dMaxQ achieves the highest throughput with approximately 30% and 230% higher throughput than ROMA and JazzyMAC respectively. It is worth mentioning that the throughput of NBC-ALGO remains low. It ranges from 10 to 30, and decreases as the number of flows increases from 35 to 50. This is due the reasons mentioned in Section 5.6.1. Specifically, it is

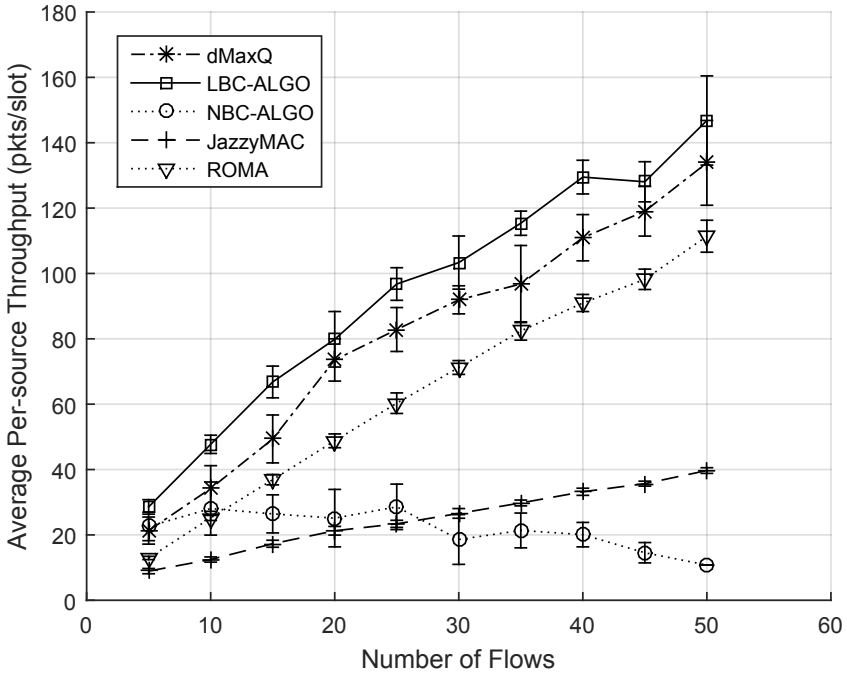


Figure 5.14: Average per-source throughput under increasing number of flows for a random topology

possible that NBC-ALGO selects transmitters that have no packets to transmit to its receivers. If packets are not served, queues will build up, which leads to reduced flow rates. As a result, the average per-source throughput decreases.

Compared to LBC-ALGO, dMaxQ produces approximately 10% lower throughput. However, the former requires queue information to be sent in real time to a central controller that runs LBC-ALGO, which may be multiple hops away; such requirement is impractical and incurs excessive delays and signaling overheads. In contrast, each node using dMaxQ exchanges information only with its one-hop neighbors. As global information, i.e., the length of all queues, is not available, the set of links activated by dMaxQ in each transmission slot is necessarily sub-optimal.

Next, to study the impact of network diameter, the number of hops between the source and destination is increased. In Figure 5.15, we see that the throughput is on average about 45% of the throughput of the single-hop case. Further, a similar 10% throughput difference between dMaxQ and the two centralized algorithms. As indicated earlier, dMaxQ may activate a sub-optimal set of links. In addition,

observe that the throughput becomes steady. This is because the transmission order of links converges onto a period. Specifically, recall that links are activated whenever their corresponding non-interfering links are in-active. We can see that these links are activated alternately; all tested link schedulers exhibit the same behavior. Critically, the activation times of links have no relationship with the number of hops. The throughput of sources thus converges onto the service rate afforded by each scheduler. We see that amongst the distributed link schedulers, the throughput when using dMaxQ is 76% and 60% higher than JazzyMAC and ROMA, respectively. When compare and contrast the performance of dMaxQ in

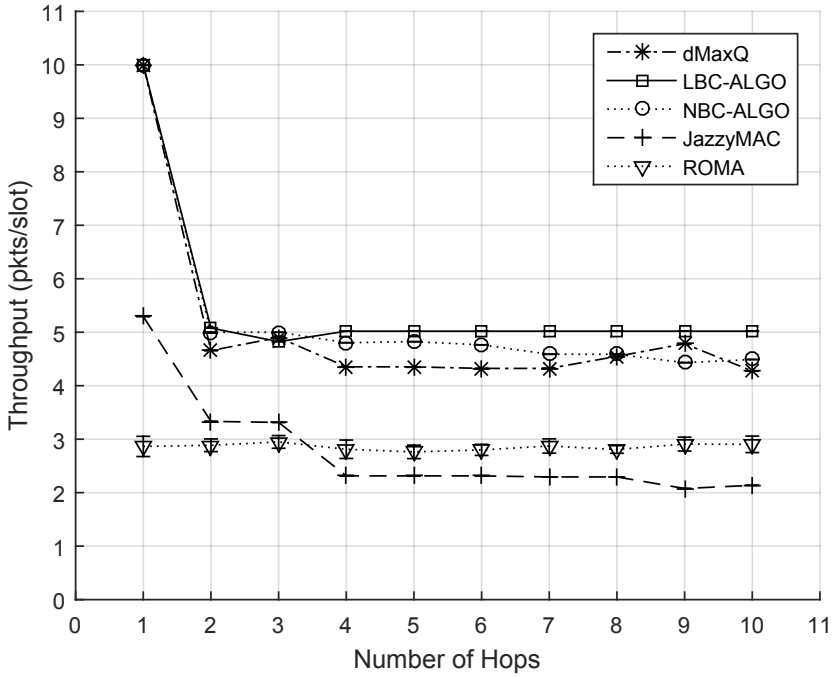


Figure 5.15: Throughput under increasing number of hops

both the multi-hop and single-hop case, we see that although in the single-hop case dMaxQ achieves 100% throughput, this percentage decreased by 10% in the multi-hop case. This is because when packets traverse multiple hops, the queue at a source will take a longer time to change and be depleted. This is particularly discernible for flows that take very long paths.

The last metric to consider is fairness. Figure 5.16 shows the Jain fairness index of each algorithm. We can see that the fairness of ROMA is lower than LBC-ALGO,

but higher than dMaxQ. This is because in each slot, ROMA randomly selects a batch of nodes to be transmitters and all their neighbors as receivers. Thus, the transmission opportunity for every link is equal. However, as shown earlier, this is at the detriment of throughput and queue stability; see Figure 5.14. This is because ROMA may *not* pick the highest weighted links.

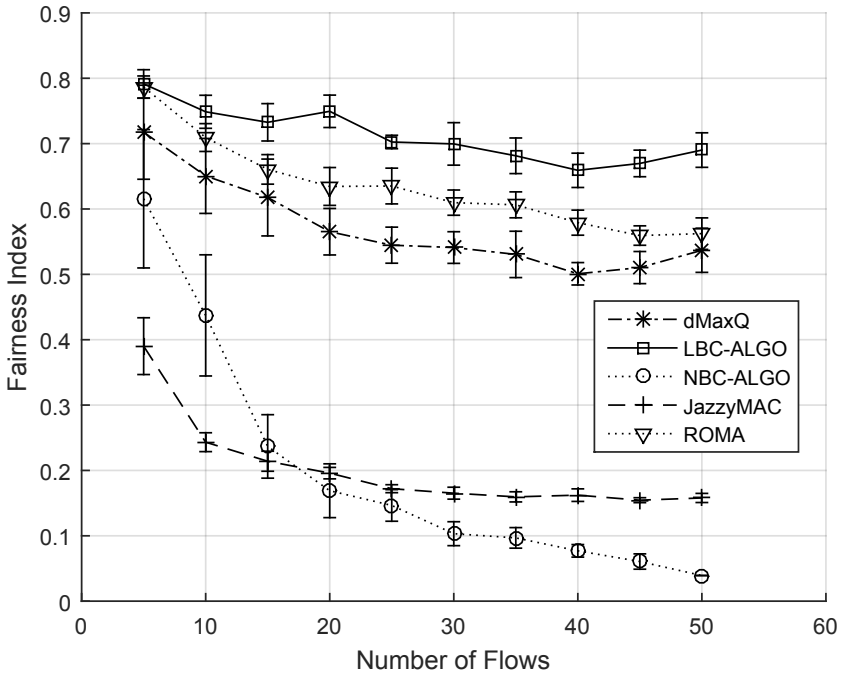


Figure 5.16: Fairness index under increasing number of flows for a random topology

5.7 Conclusion

This chapter has developed and studied a queue-aware link scheduler for MTR WMNs. In particular, the key innovation is dMaxQ, a distributed link scheduler that activates links based on queue length or queue differential. The results show that dMaxQ significantly outperforms other distributed state-of-the-art approaches. Specifically, the throughput of dMaxQ achieves 100% and 90% that of the centralized approach for single-hop and multi-hop cases respectively. Furthermore, its throughput is two to nine times higher than that of JazzyMAC, and up to 40% more than that of ROMA.

Chapter 6

Conclusion

This thesis has investigated TDMA-based link scheduling algorithms for WMNs. The aim is to generate an interference-free schedule that assigns the transmission rights for each time slot to links in accordance to their traffic demand. Such a schedule determines how efficiently the channel is being used; that is, it determines a WMN's network capacity. However, Internet services operating over *contemporary* WMNs suffer from inadequate capacity, low throughput, and long delays due to poor coordination among transmitting nodes. To this end, a key novelty with respect to existing works is that this thesis considers a distinctly different interference model whereby nodes are able to transmit or receive on all their links, called MTR capability. Consequently, a major research issue is to design a link scheduler that exploits the full advantage of MTR. The fundamental idea is to schedule as many links simultaneously as possible in order to reach the maximum capacity. To this end, this thesis contributes with three novel link schedulers.

Chapter 3 defines a superframe or a link schedule as consisting of a number of time slots that afford links transmission opportunities. The problem is to derive the shortest feasible superframe whereby every link can be activated at least once for the period of its assigned air-time. The problem is significant because a short superframe ensures low end-to-end delays, and increases network capacity.

Unfortunately, this problem is NP-hard. Thus, the majority of studies are based on heuristics. Specifically, the proposed scheduler, called A-TxRx, greedily adds links whenever a link finishes transmission. As a result, unlike previous schedulers, links can start transmitting/receiving as soon as there is no conflict. The performance of A-TxRx is evaluated in various network configurations, and compared against two state-of-the-art approaches: 2P and JazzyMAC. The results show A-TxRx outperform these algorithms significantly, especially when the network becomes denser. Specifically, the superframe length of A-TxRx is typically less than half of 2P and JazzyMAC, with 60% more concurrently transmitting links.

In practice, distributed schedulers are preferred due to their scalability, smaller connection setup delays and smaller management overheads. Henceforth, Chapter 4 proposes a distributed scheduler called PCP-TDMA that considers MTR nodes. PCP-TDMA aims to gradually shorten the initial TDMA schedule using only one-hop local information. This is significant because it eliminates the need to send topological information to a central controller, and to disseminate the computed schedule to each node, and thus saves enormous signaling overheads. To be specific, nodes using PCP-TDMA first improve their reserved slots by moving the activation time of their links to the beginning of the current superframe. Then nodes remove all idle slots at the end to shorten the superframe. In addition, PCP-TDMA considers topology changes and provides simple solutions to re-adjust the current superframe when existing or new nodes leave or join the network. The results show that PCP-TDMA approximates the performance of its centralized counterpart, and significantly outperforms state-of-the-art distributed approaches.

In order to avoid large delay and packet loss, Chapter 5 considers queue stability, i.e., queue lengths remain finite at all times, using a distributed, throughput optimal, link scheduler. Throughput optimality means a scheduler can achieve the largest possible capacity region. In this regard, a key indicator that a scheduler has superior performance is that it supports a high load whilst its queue lengths remain short. In this respect, this thesis considers the case where a central controller has

global information of all queue information. The controller then iteratively activates the set of heaviest, non-interfering links in each time slot. This so called centralized algoritm called LBC-ALGO serves only as a benchmark because the central controller is not achievable in practice due to delays and signaling overheads. To this end, a distributed protocol called dMaxQ is designed whereby each node only exchanges queue information with its one-hop neighbors. This is significant because dMaxQ does not require a central controller and does not have excessive communication overheads associated with queue length collection. dMaxQ is evaluated in both one-hop and multi-hop scenarios. The one-hop traffic case studies dMaxQ in scenarios that are independent of any routing or transport protocols. Consequently, it only focuses on dMaxQ's ability to keep queue stable and maximize throughput. The performance in terms of queue length, delay and capacity region are measured and compared against that obtained by the theoretical, centralized scheduler. The fact that dMaxQ achieves the same capacity region as LBC-ALGO in all one-hop scenarios indicates that dMaxQ is throughput optimal. Then in multi-hop cases where traffic demands span multiple consecutive links, queue differential instead of queue backlog is used as weight. Numerical results show that dMaxQ produces approximately 40% and 230% higher throughput than distributed scheduler ROMA and JazzyMAC, respectively.

There are a number of possible future works. For instance, according to the simulation results in Chapter 3, it can be seen that the computation time of A-TxRx increases exponentially when the number of nodes or degree of nodes increases. Thus, one future work is to find a solution to address this issue. Another example is that in Chapter 5, dMaxQ iteratively selects links to transmit. Therefore, an immediate future work is to study whether the number of iterations carried out by dMaxQ can be reduced further using fewer signaling messages. Apart from that, it will be interesting to determine whether dMaxQ can be used to guarantee the QoS requirements of flows in addition to guaranteeing queue stability. Moreover, this thesis assumes that all nodes are equipped with sufficient radios to establish a link

to each of its neighbors. Thus, one possible future research direction is to investigate scenarios where nodes have a limited number of directional antennas for a node to communicate with all neighbors. Lastly, a key assumption of this thesis is half-duplex wireless mesh nodes. Therefore, another possible direction is to consider full-duplex MTR WMNs. Specifically, it will be interesting to reconsider the problems addressed in this thesis but instead nodes are allowed concurrent transmissions *and* receptions over the *same* frequency.

Bibliography

[1] "Aruba networks," Sep 12 2013. http://arubanetworks.com/.

[2] X. Wang, "Wireless mesh networks," *Journal of Telemedicine and Telecare*, vol. 14, pp. 401–403, Dec 2008.

[3] I. Akyildiz and X. Wang, "A survey on wireless mesh networks," *IEEE Communications Magazine*, vol. 43, pp. S23–S30, Sep 2005.

[4] H. Aoki, S. Takeda, K. Yagyu, and A. Yamada, "IEEE 802.11 s wireless LAN mesh network technology," *NTT DoCoMo Technical Journal*, vol. 8, pp. 13–21, Sep 2006.

[5] M. Lee, J. Zheng, Y.-B. Ko, and D. Shrestha, "Emerging standards for wireless mesh technology," *IEEE Wireless Communications*, vol. 13, pp. 56–63, Apr 2006.

[6] E. Shihab, L. Cai, F. Wan, A. Gulliver, and N. Tin, "Wireless mesh networks for in-home IPTV distribution," *IEEE Network*, vol. 22, pp. 52–57, Jan 2008.

[7] M. J. Lee, R. Zhang, J. Zheng, G. S. Ahn, C. Zhu, T. R. Park, S. R. Cho, C. S. Shin, and J. S. Ryu, "IEEE 802.15.5 WPAN mesh standard-low rate part: Meshing the wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 28, pp. 973–983, Sep 2010.

[8] "An introduction to wireless mesh networking," Mar 2005. www.firetide.com.

[9] J. Ishmael, S. Bury, D. Pezaros, and N. Race, "Deploying rural community wireless mesh networks," *IEEE Internet Computing*, vol. 12, pp. 22–29, July 2008.

[10] P. Gupta and P. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, pp. 388–404, Mar 2000.

[11] C. Joo and N. B. Shroff, "Local greedy approximation for scheduling in multi-hop wireless networks," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 414–426, Mar 2012.

[12] E. Modiano, D. Shah, and G. Zussman, "Maximizing throughput in wireless networks via gossiping," *ACM SIGMETRICS Performance Evaluation Review*, vol. 34, pp. 27–38, Jun 2006.

[13] K. Chin, "A new link scheduling algorithm for concurrent Tx/Rx wireless mesh networks," in *IEEE ICC*, (Beijing, China), pp. 3050–3054, May 2008.

[14] 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 *ACM NSDI*, (Cambridge, MA,USA), pp. 87–100, Apr 2007.

[15] P. Dutta, S. Jaiswal, and R. Rastogi, "Routing and channel allocation in rural wireless mesh networks," in *IEEE INFOCOM*, (Washington, DC, USA), pp. 598–606, May 2007.

[16] B. Raman and K. Chebrolu, "Design and evaluation of a new MAC protocol for long-distance 802.11 mesh networks," in *ACM MOBICOM*, (Cologne, Germany), pp. 156–169, Aug 2005.

[17] M. Jain, J. I. Choi, T. Kim, D. Bharadia, S. Seth, K. Srinivasan, P. Levis, S. Katti, and P. Sinha, "Practical, real-time, full duplex wireless," in *ACM MOBICOM*, (New York, NY, USA), pp. 301–312, ACM, Sep 2011.

[18] R. Mudumbai, S. Singh, and U. Madhow, "Medium access control for 60 GHz outdoor mesh networks with highly directional links," in *IEEE INFOCOM*, (Rio de Janeiro, Brazil), pp. 2871–2875, Apr 2009.

[19] S. Chu and X. Wang, "Opportunistic and cooperative spatial multiplexing in MIMO ad hoc networks," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1610–1623, Oct 2010.

[20] "Argos: Practical many-antenna base stations," Apr 22 2013. http://argos.rice.edu/.

[21] D. Tse and P. Viswanath, *Fundamentals of wireless communication*. Cambridge university press, 2005.

[22] R. Nelson and L. Kleinrock, "Spatial TDMA: a collision-free multihop channel access protocol," *IEEE Transactions on Communications*, vol. 33, pp. 934–944, Sep 1985.

[23] 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.

[24] V. V. Vazirani, *Approximation algorithms*. Springer Science & Business Media, 2013.

[25] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1936–1948, Aug 1992.

[26] S. Nedevschi, R. Patra, S. Surana, S. Ratnasamy, L. Subramanian, and E. Brewer, "An adaptive, high performance MAC for long-distance multihop wireless networks," in *ACM MOBICOM*, (New York, NY, USA), pp. 259–270, Sep 2008.

[27] L. Bao and J. Garcia-Luna-Aceves, "Transmission scheduling in ad hoc networks with directional antennas," in *ACM MOBICOM*, (Atlanta, Georgia, USA), pp. 48–58, Sep 2002.

[28] V. K. Garg and J. E. Wilkes, *Wireless and personal communications systems*. Prentice Hall PTR, 1996.

[29] 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, pp. 2301 – 2312, Jan 2013.

[30] P. Cappanera, L. Lenzini, A. Lori, G. Stea, and G. Vaglini, "Link scheduling with end-to-end delay constraints in wireless mesh networks," in *IEEE WoWMoM*, (Kos, Greece), pp. 1–9, Jun 2009.

[31] A. Capone, I. Filippini, and F. Martignon, "Joint routing and scheduling optimization in wireless mesh networks with directional antennas," in *IEEE ICC*, (Beijing, China), pp. 2951–2957, May 2008.

[32] S. Das, H. Pucha, D. Koutsonikolas, Y. Hu, and D. Peroulis, "DMesh: incorporating practical directional antennas in multichannel wireless mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 2028–2039, Nov 2006.

[33] P. Krishnamurthy and S. Krishnamurthy, "Use of smart antennas in Ad Hoc networks," in *Ad Hoc Networks*, pp. 197 –228, Springer US, Jan 2005.

[34] D. Hao and D. P. Liu, "A distributed TDMA-based MAC protocol for ad hoc

networks with directional antennas," in *International Conference on Wireless Communications Signal Processing*, (Huangshan, China), pp. 1–6, Oct 2012.

[35] A. Kumar and R. Tripathi, "Effective resource utilization in wireless mesh networks using smart antenna," in *WOCN*, (Colombo, Sri Lanka), pp. 1–5, Sep 2010.

[36] S. Atmaca, C. Ceken, and I. Erturk, "Capacity enhancement in wireless networks using directional antennas," *World Enformatika Society, Transactions on Engineering, Computing and Technology*, vol. 1, pp. 174 –179, 2006.

[37] V. Ramamurthi, A. Reaz, S. Dixit, and B. Mukherjee, "Link scheduling and power control in wireless mesh networks with directional antennas," in *IEEE ICC*, (Beijing, China), pp. 4835–4839, May 2008.

[38] M. Garache, *Multihop wireless networks with advanced antenna systems: an alternative for rural communication.* Phd. dissertation, Communication Systems, School of ICT, KTH, Stockholm, Sweden, 2008.

[39] T. Paso, J. P. Makela, and J. Iinatti, "TDMA slot borrowing scheme utilizing smart antennas in ad hoc networks," in *MILCOM*, (San Jose, CA, USA), pp. 1930–1935, Oct 2010.

[40] A. Spyropoulos and C. Raghavendra, "Energy efficient communications in ad hoc networks using directional antennas," in *IEEE INFOCOM*, vol. 1, (New York, NY, USA), pp. 220–228, Jun 2002.

[41] T. Yoo and A. Goldsmith, "On the optimality of multiantenna broadcast scheduling using zero-forcing beamforming," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 528–541, Mar 2006.

[42] J. Zhang and X. Jia, "Capacity analysis of wireless mesh networks with omni or directional antennas," in *IEEE INFOCOM*, (Honolulu, Hawaii, USA), pp. 2881–2885, Apr 2009.

[43] S. Yi, Y. Pei, and S. Kalyanaraman, "On the capacity improvement of ad hoc wireless networks using directional antennas," in *ACM MOBIHOC*, (Annapolis, MD, USA), pp. 108–116, Jun 2003.

[44] H. Dai, K. Ng, M. Li, and M. Wu, "An overview of using directional antennas in wireless networks," *International Journal of Communication Systems*, pp. 1–39, Apr 2011.

[45] M. Takai, J. Martin, R. Bagrodia, and A. Ren, "Directional virtual carrier sensing for directional antennas in mobile Ad Hoc networks," in *ACM MOBIHOC*, (Lausanne, Switzerland), pp. 183–193, ACM, Jun 2002.

[46] R. Choudhury and N. Vaidya, "Deafness: a MAC problem in ad hoc networks when using directional antennas," in *IEEE ICNP*, (Berlin, Germany), pp. 283–292, Oct 2004.

[47] H. N. Dai, K. W. Ng, and M. Y. Wu, "A busy-tone based mac scheme for wireless ad hoc networks using directional antennas," in *IEEE GLOBECOM*, (Washington, DC, USA), pp. 4969–4973, Nov 2007.

[48] A. El Masri, L. Khoukhi, and D. Gaiti, "A TDMA-based MAC protocol for wireless mesh networks using directional antennas," in *International Conference on Communication Theory, Reliability, and Quality of Service*, (Budapest, Hungary), pp. 95–98, Apr 2011.

[49] A. S. Tanenbaum, *Computer Networks*. New Jersey: Prentice Hall, fourth edition ed., 2002.

[50] J. Zou and D. Zhao, "Real-time CBR traffic scheduling in IEEE 802.16-based wireless mesh networks," *Wireless Networks*, vol. 15, pp. 65–72, Jan 2009.

[51] D. Ghosh, A. Gupta, and P. Mohapatra, "Admission control and interference-aware scheduling in multi-hop WiMAX networks," in *IEEE International Conference on Mobile Adhoc and Sensor Systems*, (Pisa, Italy), pp. 1–9, Oct 2007.

[52] M. Alicherry, R. Bhatia, and L. E. Li, "Joint channel assignment and routing for throughput optimization in multiradio wireless mesh networks," *IEEE Journal on Selected Areas in Communications*, vol. 24, pp. 1960–1971, Nov 2006.

[53] B. Gendron, A. Hertz, and P. St-Louis, "On edge orienting methods for graph coloring," *Journal of Combinatorial Optimization*, vol. 13, pp. 163–178, Feb 2007.

[54] P. Dutta, S. Jaiswal, D. Panigrahi, and R. Rastogi, "A new channel assignment mechanism for rural wireless mesh networks," in *IEEE INFOCOM Mini-Conference*, (Phoenix, AZ, USA), pp. 246–250, Apr 2008.

[55] J. Malinen, "Host AP driver for intersil prism2/2.5/3, hostapd, and WPA supplicant." https://w1.fi/.

[56] H. Blcskei, D. Gesbert, C. B. Papadias, and A.-J. v. d. Veen, *Space-Time Wireless Systems: From Array Processing to MIMO Communications*. Cambridge University Press, Jun 2006.

[57] D. Gesbert, M. Kountouris, R. Heath, C. Chae, and T. Salzer, "Shifting the MIMO paradigm," *Signal Processing Magazine*, vol. 24, pp. 36–46, Sep 2007.

[58] M. Bahadori and K. Psounis, "On the performance of multiuser MIMO mesh networks," Tech. Rep. CENG-2010-7, Univ. of South Carlifonia, Los Angeles, CA, USA, Aug 2010.

[59] K. Huang, J. Andrews, and R. Heath, "Performance of orthogonal beamforming for SDMA with limited feedback," *IEEE Transactions on Vehicular Technology*, vol. 58, pp. 152–164, Jan 2009.

[60] M. Costa, "Writing on dirty paper (Corresp.)," *IEEE Transactions on Information Theory*, vol. 29, pp. 439–441, May 1983.

[61] Q. Spencer, C. Peel, A. Swindlehurst, and M. Haardt, "An introduction to the multi-user MIMO downlink," *IEEE Communications Magazine*, vol. 42, pp. 60–67, Oct 2004.

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

[63] J. Qiao, L. X. Cai, X. Shen, and J. W. Mark, "STDMA-based scheduling algorithm for concurrent transmissions in directional millimeter wave networks," in *IEEE ICC*, (Ottawa, Cananda), pp. 5221–5225, June 2012.

[64] K. Chin, S. Soh, and C. Meng, "A novel spatial TDMA scheduler for concurrent transmit/receive wireless mesh networks," in *IEEE AINA*, (Perth, WA, Australia), pp. 1–10, Apr 2010.

[65] H.-Y. Loo, S. Soh, and K. W. Chin, "On improving capacity and delay in multi Tx/Rx wireless mesh networks with weighted links," in *Asia-Pacific Conference on Communications*, (Denpasar, Indonesia), pp. 12–17, Aug 2013.

[66] H. Wang, K.-W. Chin, S. Soh, and R. Raad, "A distributed maximal link scheduler for multi Tx/Rx wireless mesh networks," *IEEE Transactions on Wireless Communications*, vol. 14, pp. 520–531, Jan 2015.

[67] K. Chin, S. Soh, and C. Meng, "A novel scheduler for concurrent Tx/Rx wireless mesh networks with weighted links," *IEEE Communications Letters*, vol. 16, pp. 246–248, Feb 2012.

[68] H. Dai, S. Liew, and L. Fu, "Link scheduling in multi-transmit-receive wireless networks," in *IEEE LCN*, (Bonn, Germany), pp. 199–202, Oct 2011.

[69] J. Ward and R. T. Compton, "High throughput slotted ALOHA packet radio networks with adaptive arrays," *IEEE Transactions on Communications*, vol. 41, pp. 460–470, Mar 1993.

[70] L. Bao and J. J. Garcia-Luna-Aceves, "A new approach to channel access scheduling for ad hoc networks," in *ACM MOBICOM*, (Roma, Italy), pp. 210–221, ACM, Jul 2001.

[71] A. Das and T. Zhu, "A reservation-based TDMA MAC protocol using directional antennas (RTDMA-DA) for wireless mesh networks," in *IEEE GLOBECOM*, (Washington, DC, USA), pp. 5102–5106, Nov 2007.

[72] S. K. Hazra and W. K. G. Seah, "Measurement-based link scheduling for maritime mesh networks with directional antennas," *International Journal of Network Management*, vol. 21, pp. 83–105, Mar 2011.

[73] I. K. Son, S. Mao, M. X. Gong, and Y. Li, "On frame-based scheduling for directional mmWave WPANs," in *IEEE INFOCOM*, (Orlando, Florida USA), pp. 2149–2157, Mar 2012.

[74] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized TDMA scheduling for wireless Ad-Hoc networks," in *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, (Florence, Italy), pp. 190–201, ACM, May 2006.

[75] S. Ramanathan, "A unified framework and algorithm for (T/F/C)DMA channel assignment in wireless networks," in *IEEE INFOCOM*, vol. 2, (Kobe, Japan), pp. 900–907, Apr 1997.

[76] J. Qiao, L. X. Cai, X. S. Shen, and J. W. Mark, "Enabling multi-hop concurrent transmissions in 60 GHz wireless personal area networks," *IEEE Transactions on Wireless Communications*, vol. 10, pp. 3824–3833, Nov 2011.

[77] C. T. Chang, C. Y. Chang, and Y. J. Lu, "Maximizing throughput by exploiting spatial reuse opportunities with smart antenna systems," in *IEEE ICC*, (Cape Town, South Africa), pp. 1–5, May 2010.

[78] L. Wang, K.-W. Chin, S. Soh, and R. Raad, "Novel joint routing and scheduling algorithms for minimizing end-to-end delays in multi Tx-Rx wireless mesh networks," *Computer Communications*, vol. 72, pp. 63 – 77, Jun 2015.

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

[80] K. Chebrolu and B. Raman, "FRACTEL: a fresh perspective on (rural) mesh networks," in *ACM Workshop on Networked systems for developing regions*, (Kyoto, Japan), pp. 8:1–8:6, Aug 2007.

[81] B. Raman and K. Chebrolu, "Experiences in using WiFi for rural internet in india," *IEEE Communications Magazine*, vol. 45, pp. 104–110, Jan 2007.

[82] H. Yu, P. Mohapatra, and X. Liu, "Dynamic channel assignment and link scheduling in multi-radio multi-channel wireless mesh networks," in *MobiQuitous*, (Philadelphia, PA), pp. 1–8, Aug 2007.

[83] C. Eklund, R. B. Marks, K. L. Stanwood, and S. Wang, "IEEE standard 802.16: a technical overview of the wirelessMAN air interface for broadband wireless access," *IEEE Communications Magazine*, vol. 40, pp. 98–107, Jun 2002.

[84] B. Raman, "Channel allocation in 802.11-based mesh networks," in *IEEE INFOCOM*, (Barcelona, Catalunya, Spain), pp. 1–10, Apr 2006.

[85] G. Sharma, R. Mazumdar, and N. Shroff, "On the complexity of scheduling in wireless networks," in *ACM MOBICOM*, (Los Angeles, CA, USA), pp. 227–238, Sep 2006.

[86] L. Tassiulas, "Linear complexity algorithms for maximum throughput in radio networks and input queued switches," in *IEEE INFOCOM*, (San Francisco, CA), pp. 533–539, Mar 1998.

[87] J.-H. Hoepman, "Simple distributed weighted matchings," Oct 2004. Preprint available at http://arxiv.org/abs/cs.DC/0410047.

[88] X. Lin and S. Rasool, "Constant-time distributed scheduling policies for Ad Hoc wireless networks," in *IEEE Conference on Decision and Control*, (San Diego, CA), pp. 231–242, Dec 2006.

[89] P. Chaporkar, "Achieving queue length stability through maximal scheduling in wireless networks," in *Proceedings of Information Theory and Applications*, (San Diego, La Jolla, CA), pp. 6–10, Feb 2006.

[90] A. Gupta, Lin.X, and R. Srikant., "Low-complexity distributed scheduling algorithms for wireless networks," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1846–1859, Dec 2009.

[91] L. Bui, S. Sanghavi, and R. Srikant, "Distributed link scheduling with constant overhead," *IEEE/ACM Transactions on Networking*, vol. 17, pp. 1467–1480, Jun 2009.

[92] B. Radunovic, C. Gkantsidis, D. Gunawardena, and P. Key, "Horizon: balancing TCP over multiple paths in wireless mesh network," in *ACM MOBICOM*, (San Francisco, California, USA), pp. 247–258, Sep 2008.

[93] A. Warrier, S. Janakiraman, S. Ha, and I. Rhee, "DiffQ: practical differential backlog congestion control for wireless networks," in *IEEE INFOCOM*, (Rio de Janeiro, Brazil), pp. 262–270, Apr 2009.

[94] J.-Y. Yoo, C. Sengul, R. Merz, and J. Kim, "Backpressure scheduling in IEEE 802.11 wireless mesh networks: Gap between theory and practice," *Computer Networks*, vol. 56, pp. 2934 – 2948, May 2012.

[95] R. Laufer, T. Salonidis, H. Lundgren, and P. Le Guyadec, "A cross-layer backpressure architecture for wireless multihop networks," *IEEE/ACM Transactions on Networking*, vol. 22, pp. 363–376, Apr 2014.

[96] D. W. Matula and L. L. Beck, "Smallest-last ordering and clustering and graph coloring algorithms," *Journal of the ACM*, vol. 30, pp. 417–427, Jul 1983.

[97] E. R. Scheinerman, "Matgraph: a MATLAB toolbox for graph theory," *Department of applied mathematics and statistics, the Johns Hopkins University, Baltimore, Maryland*, pp. 1–7, 2008.

[98] N.-C. Wang, Y.-F. Huang, and J.-C. Chen, "A stable weight-based on-demand routing protocol for mobile ad hoc networks," *Information Sciences*, vol. 177, pp. 5522 – 5537, May 2007.

[99] F. P. Kelly, A. K. Maulloo, and D. K. H. Tan, "Rate control for communication networks:shadow prices, proportional fairness and stability," *Journal of the Operational Research Society*, vol. 49, pp. 237–252, Mar 1998.