Link Scheduling Algorithms For In-Band Full-Duplex Wireless Networks

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

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Yifei Ren
Bachelor of Engineering (Telecommunications)
School of Electrical, Computer and Telecommunications Engineering
August, 2019
Statement of Originality

I, Yifei Ren, declare that this thesis, submitted in partial fulfillment 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

Yifei Ren
August, 2019
Abstract

In the last two decades, wireless networks and their corresponding data traffic have grown significantly. This is because wireless networks have become an indispensable and critical communication infrastructure in a modern society. An on-going challenge in communication systems is meeting the continuous increase in traffic demands. This is driven by the proliferation of electronic devices such as smartphones with a WiFi interface along with their bandwidth intensive applications. Moreover, in the near future, sensor devices that form the Internet of Things (IoTs) ecosystem will also add to future traffic growth.

One promising approach to meet growing traffic demands is to equip nodes with an In-band-Full-Duplex (IBFD) radio. This radio thus allows nodes to transmit and receive data concurrently over the same frequency band. Another approach to increase network or link capacity is to exploit the benefits of Multiple-Input-Multiple-Output (MIMO) technologies; namely, (i) spatial diversity gain, which improves Signal-to-Noise Ratio (SNR) and thus has a direct impact on the data rate used by nodes, and (ii) spatial multiplexing gain, whereby nodes are able to form concurrent links to neighbors.

This thesis aims to develop novel algorithms to schedule links from nodes with IBFD or MIMO technologies. These link schedulers play a critical role in determining the network capacity of a wireless network. A short schedule means a link can
be activated frequently, and thus it has a high link capacity. Apart from that, these algorithms determine whether a set of links can co-exist together. In particular, a high spatial reuse or number of concurrent links correspond to a high network capacity. In this respect, link schedulers must ensure co-existing links do not suffer from severe interference, and links are able to operate using a high data rate. In the case of MIMO-capable nodes, they must allocate their antenna elements efficiently to support a high number of data streams as well as cancel any interference. Lastly, these link schedulers also need to consider the amount of data to be transmitted by each node. This is made more challenging if traffic arrival and interference are random, especially when nodes have imperfect state of knowledge.

This thesis makes three contributions. The first contribution concerns minimizing the transmission completion time of a given set of links. These links have varying amounts of data to transmit, and the problem at hand is to determine the start and end time of links such that the end time of the last scheduled link is minimized. The key challenge in the said problem is that links may interfere with one another. In particular, if there are many active links, then they may need to use a low data rate due to excessive interference. Consequently, their transmission time will increase. On the other hand, if links are scheduled one after another, although there is no interference, the spatial reuse or network capacity will be low. To this end, this thesis proposes three heuristic algorithms to minimize completion time. They determine the links to be scheduled whenever a link finishes transmission and also their data rate. To select links, they use the concept of ‘affectedness’, which indicates whether a link can be activated concurrently with a given set of links. The simulation results show that the overall completion time can be reduced by about 40% as compared to prior solutions.

The second contribution considers random channel gains when scheduling links. This is significant because data rate must be chosen appropriately according to Signal-to-Interference-Plus-Noise Ratio (SINR), especially when multiple links are scheduled together. Hence, in practice, nodes require expensive channel estimation
in order to determine the channel state. However, wireless channel state may change quickly data transmission. In addition, an on-going transmission may experience a collision. To this end, this thesis considers link scheduling with imperfect channel state information. Nodes use a reinforcement learning approach, namely hierarchical Q-Learning, to learn and select the most suitable link and data rate pairs. Advantageously, nodes are able to adapt to varying channel condition and maintain a high throughput without any channel state information. In this regard, this thesis is the first to study a link scheduler that takes advantage of machine learning and IBFD technologies. The results show that the proposed distributed Q-learning based scheduling algorithm achieves an average throughput that is 200% higher than Carrier Sense Multiple Access (CSMA), and up to 300% higher than Time Division Multiple Access (TDMA).

The last contribution concerns link scheduling, allocation of antenna elements, random channel gains and traffic arrival in a Wireless Local Area Network (WLAN). Nodes are both IBFD and MIMO capable. A key challenge is allocating the antenna elements of nodes efficiently. In particular, nodes can use their antenna elements for data transmissions or to cancel interference from their own transmissions as well as interfering transmissions from neighboring cells. Another challenging aspect is random traffic arrival. The problem at hand is to schedule a bi-directional or a relay link in each time slot. To do this, an access point needs to consider the current state of the WLAN, where the state corresponds to the number of interfering streams experienced by itself and associated clients and also queue lengths. The problem is modeled as a Markov Decision Process (MDP) where in each state, the problem is to select the action or clients and antenna element allocation that yield the highest reward. This thesis contains two heuristic antenna allocation algorithms, and employs the Q-Learning algorithm to derive the best action for each state. The results show the proposed algorithm is able to activate on average 60% more data streams as compared to polling-based methods while maintaining low packets drops.
Acknowledgments

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Kwan-Wu Chin, for being the best mentor I ever met. Being his student is a wonderful experience. Without his selfless guidance, inspiring ideas, encouragement, countless contributed time and hard work, it would be impossible for me to finish my PhD degree and publish my research. I sincerely appreciate his help in improving my writing skills. His detailed editing and comments on my papers not only helped me publish my papers but also benefit me in the future. I would also like to thank him for his encouragement when I met a rejection for the first time.

Special thanks to Dr Sieteng Soh and my co-supervisor Dr Raad Raad, for their contributions and help in refining my research work.

Sincere thanks to the reviewers and editors of journal papers for their constructive and useful comments, which have further improved the quality of my works.

Special thanks to my friends and research mates, Mr Dawei Gao and Miss Yishun Wang, for the countless joyful and inspiring conversations, and their help when I encountered various technical difficulties.

Finally, I would like to thank my parents. Without their hard work, love, encouragement, it would be impossible for me to study abroad in the first place. Although they faced significant financial hardship after I finished my Bachelor degree, they still chose to believe in my abilities and allowed me to continue my study. They
have always been a great comfort to me and have always motivated me to become a better person.
# Contents

Abstract

Acknowledgments

Abbreviations

1 Introduction

1.1 Problem Space and Motivation

1.1.1 Minimizing Completion Time

1.1.2 Varying Channel Condition

1.1.3 Random Packet Arrivals and Interference

1.2 Contributions

1.2.1 Minimizing completion time in IBFD WLANs

1.2.2 A Distributed Q-Learning Based Link Scheduler

1.2.3 Scheduling Packets over the DoF Model

1.3 Publications

1.4 Thesis Structure

2 Literature Review

2.1 Physical Layer Technologies

2.1.1 Multiple Channels
List of Figures

1.1 An example network with a) a single half-duplex transmission, and b) a relay full-duplex transmission, where node-4’s transmission causes interference at node-1. ................................. 2
1.2 Example scenarios of full-duplex transmissions. ......................... 3
1.3 Half-duplex schedule example and full-duplex schedule example. . . . 5
1.4 A dense WLAN with a central controller, and all devices operate on the same frequency. Also shown are full-duplex links, as indicated by double headed arrows. ................................. 7
1.5 Example schedules. .................................................. 8
1.6 An example of distributed WLAN. .................................... 9
1.7 An centralized WLAN under interfering streams from other cell. . . . 10
2.1 Collisions between flows in full-duplex transmissions. ..................... 28

3.1 A dense WLAN consists two cells. Two APs are connected to a central controller. Full-duplex links are indicated by a double headed blue arrow. ................................................. 41
3.2 Interference scenarios: (a) self, (b) cross, (c) exogenous. ................ 45
3.3 An example link schedule. ............................................. 46
3.4 An example WLAN with multiple APs. Strong and weak interference are denoted by thick and thin dotted lines, respectively. .............. 50
3.5 Result of Algorithm 1. As a comparison, also shown is a schedule whereby links transmit one after another; i.e., no concurrent trans-
missions. This schedule ends at $T_e$. .................................. 51
3.6 Result of Algorithm 2. This figure also shows a schedule without concurrent transmissions that ends at time $T_e$. .................... 54
3.7 Average completion time reduction versus node density. ............ 62
3.8 Maximum completion time reduction versus node density. ........... 63
3.9 Minimum completion time reduction versus node density. ........... 63
3.10 Computation time versus node density. .............................. 66
3.11 Average completion time reduction versus transmission power. .... 67
3.12 Maximum completion time reduction versus transmission power. .. 68
3.13 Minimum completion time reduction versus transmission power. .. 69
3.14 Average completion time reduction versus SINR threshold $\beta$. ... 70
3.15 Maximum completion time reduction versus SINR threshold $\beta$. .. 70
3.16 Minimum completion time reduction versus SINR threshold $\beta$ .... 71
4.1 Reinforcement learning versus hierarchical reinforcement learning. .. 78
4.2 A hierarchy of actions. .................................................. 80
4.3 Three nodes transmit an NTS frame according to their ID. .......... 82
4.4 Topology with five nodes. ............................................. 91
4.5 Number of iterations versus the values of learning rate $\alpha$. ........ 92
4.6 Learning rate $\alpha^1$, $\alpha^2$ and $\alpha^3$ versus the number of iterations. .... 92
4.7 Discount factor $\gamma$ versus the average throughput. ............... 94
4.8 Cumulative average reward for choosing a data rate versus the number of iterations for different discount factor $\gamma$. ............... 94
4.9 Topologies with different number of nodes. .......................... 95
4.10 Average throughput over different topologies. ....................... 96
<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.11</td>
<td>Number of NTS frames with flag = 1 sent in the training stage.</td>
<td>97</td>
</tr>
<tr>
<td>4.12</td>
<td>Average number of NTS frames with flag = 1 sent in the simulation stage.</td>
<td>98</td>
</tr>
<tr>
<td>4.13</td>
<td>Training time.</td>
<td>99</td>
</tr>
<tr>
<td>4.14</td>
<td>Topology with six nodes located in two cells.</td>
<td>99</td>
</tr>
<tr>
<td>4.15</td>
<td>Average throughput achieved by algorithms in a two cells topology.</td>
<td>100</td>
</tr>
<tr>
<td>4.16</td>
<td>Average throughput over varying channel condition.</td>
<td>101</td>
</tr>
<tr>
<td>5.1</td>
<td>A WLAN with MIMO and full duplex capability. MIMO full duplex link is shown</td>
<td>103</td>
</tr>
<tr>
<td></td>
<td>by double head blue arrows. Each node is experiencing interfering streams</td>
<td></td>
</tr>
<tr>
<td></td>
<td>from neighboring cells, shown by orange arrows.</td>
<td></td>
</tr>
<tr>
<td>5.2</td>
<td>Average data streams and packet drops after training learning Rate $\alpha$.</td>
<td>116</td>
</tr>
<tr>
<td>5.3</td>
<td>Average data streams and packet drops after training with different weight</td>
<td>117</td>
</tr>
<tr>
<td></td>
<td>$\sigma$.</td>
<td></td>
</tr>
<tr>
<td>5.4</td>
<td>Average data streams in topologies with varying number of clients.</td>
<td>118</td>
</tr>
<tr>
<td>5.5</td>
<td>Average packet drops in topologies with varying number of clients.</td>
<td>119</td>
</tr>
<tr>
<td>5.6</td>
<td>Average data streams under various maximum queue sizes.</td>
<td>120</td>
</tr>
<tr>
<td>5.7</td>
<td>Average packet drops under various maximum queue sizes.</td>
<td>120</td>
</tr>
<tr>
<td>5.8</td>
<td>Average data streams under different packets arrival rates.</td>
<td>122</td>
</tr>
<tr>
<td>5.9</td>
<td>Average packet drops under different packets arrival rates.</td>
<td>122</td>
</tr>
</tbody>
</table>
## List of Tables

2.1 A comparison of full duplex MACs .......................... 34
2.2 A comparison of learning based MACs ......................... 38
3.1 SINR thresholds and their corresponding data rate [1] .... 44
4.1 SINR thresholds and their corresponding data rate [1] .... 75
4.2 Symbols and Description ..................................... 75
4.3 Simulation parameters ....................................... 89
5.1 Symbols and Description ..................................... 107
5.2 Simulation parameters ....................................... 114
Abbreviations

SNR  Signal-to-Noise Ratio
SINR Signal-to-Interference-plus-Noise Ratio
MIMO Multiple Input Multiple Output
IBFD In-Band Full Duplex
DoF Degree of Freedom
WLAN Wireless Local Area Network
BSS Basic Service Set
AP Access Point
CSMA Carrier Sense Multiple Access
TDMA Time Division Multiple Access
FDMA Frequency Division Multiple Access
CDMA Code Division Multiple Access
RTS Request To Send
CTS Clear To Send
OFDMA Orthogonal Frequency Division Multi Access
DBTMA Dual Busy Tone Multi Access
MU-MIMO Multi-Users MIMO
CSI Channel State Information
SIC Successive Interference Cancellation
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADC</td>
<td>Analog-Digital Converter</td>
</tr>
<tr>
<td>FD CTS</td>
<td>Full Duplex Clear To Send</td>
</tr>
<tr>
<td>SNAV</td>
<td>Set Network Allocation Vector</td>
</tr>
<tr>
<td>DSNAV</td>
<td>Destination Set Network Allocation Vector</td>
</tr>
<tr>
<td>MDP</td>
<td>Markov Decision Process</td>
</tr>
<tr>
<td>POMDP</td>
<td>Partially Observable Markov Decision Process</td>
</tr>
<tr>
<td>MAC</td>
<td>Medium Access Control</td>
</tr>
<tr>
<td>RL</td>
<td>Reinforcement Learning</td>
</tr>
</tbody>
</table>
In a wireless network, the *interference* experienced by nodes determines the achievable network or link capacity. This is because a high data rate requires a corresponding high Signal-to-Interference-plus-Noise Ratio (SINR). Consequently, a *link scheduler* is a fundamental component of a wireless network as its role is to determine the maximum number of links that can be scheduled or activated concurrently without causing excessive interference to one another [2]. As an example, consider Figure 1.1(a). If only node-2 is transmitting to node-1, then the transmission is only influenced by ambient noise in addition to path loss. The data rate of this transmission will be limited by its Signal-to-Noise ratio (SNR). However, if there is interference, as shown by the scenario in Figure 1.1(b) where node-4 is also transmitting to node-2, then the received signal at node-1 also includes the interference caused by node-4. In this case, the data rate of node-2 and node-1 will be interference limited; consequently, the resulting data rate will be lower as compared to the data rate of the transmission between node-4 and node-2. If node-3 is also trying to transmit to node-1 at the same time, the reception at node-1 will fail completely because the signal from node-3 and node-2 interferes strongly with each other.

To date, researchers have proposed many strategies to minimize interference. Some of which include:
1. **Multiple channels.** This reduces interference by scheduling links into multiple channels. In particular, it is necessary to have a channel assignment strategy to ensure interfering links are assigned an orthogonal or non-overlapping channel or to ensure a channel contains minimal interfering links. This ensures nodes are able to have contention-free transmissions [3]. Example works include [4–7].

2. **Directional or adaptive array antennas.** Unlike an omni-directional antenna, which radiates power equally in all directions, a directional antenna or adaptive array radiates power in one specific direction. Consequently, a directional antenna has a higher gain, coverage and connectivity. Advantageously, its use leads to higher spatial reuse because of lower interference to neighbour nodes [8]. Example works include [9], [10] and [11–13].

3. **Multiple Input Multiple Output (MIMO) [14].** Nodes equipped with MIMO technologies have two types of advantages: spatial diversity gain and spatial multiplexing gain. The former refers to its ability to transmit the same data or symbol on multiple antenna elements. This improves the Signal-to-Noise Ratio (SNR) at the receiver, and thus MIMO overcomes severe fading and improves reliability [15]. The latter MIMO feature allows a node to transmit different
Recently, researchers have shown the possibility of full-duplex communications over the same frequency; aka In-band Full-Duplex (IBFD) [24]. An example is shown in Figure 1.1 b) where node-2 is transmitting while receiving. Full-duplex technology has a long history since it was first implemented in continuous wave radar systems in the 1940s [25]. IBFD has received strong interests in both academia and industry as it has the potential to increase network capacity [26]. For example, works such as [27] and [28] have shown that although IBFD cannot directly double the network capacity of current network systems, it can achieve an average gain of about 1.5 in terms of capacity as compared to current half-duplex communication systems.

IBFD radios support two communication scenarios. Referring to Figure 1.2(a), we see a bi-directional full-duplex transmission between node A and B. In Figure 1.2(b), we see a relay-transmission between nodes A, B and C. Relay transmissions can be divided into (i) destination-based full-duplex transmission mode, where the middle node, i.e., node-B, transmits first, or (ii) source-based full-duplex transmission mode, where the middle node starts to receive first.
1.1 Problem Space and Motivation

This thesis aims to design link schedulers for nodes equipped with an IBFD radio. The main task of these link schedulers is to determine the transmission time of each link and also the set of links that can transmit simultaneously without causing too much interference to one another. An example link schedule that assumes nodes have a half-duplex or IBFD radio is shown in Figure 1.3. In schedule b), the network capacity is about double as compared to schedule a) because IBFD allows two links to be activated at the same time. Consequently, the completion time of schedule b) is shorter than schedule a).

In the foregone example, ensuring links have the required SINR is critical. This is because if the SINR is low but the transmitter transmits at a high data rate, the receiver cannot decode the received message. To see how a link scheduler plays a critical role, consider the schedule b) shown in Figure 1.3. Note that the activation time of all links in schedule b) has been increased because all links have to use a lower data rate due to the interference between two concurrently activated links.

To date, researchers have considered a number of performance metrics when scheduling links. Of interest is the schedule length, which is defined as the number of transmission slots required to afford each link at least one transmission opportunity. A short schedule ensures links are able to transmit frequently, and thus, have a high link capacity. For example, if a link has a theoretical capacity of 1 Mb/s, and it is only activated for 0.5 seconds by a link scheduler, then its link capacity is only 0.5 Mb/s. Another metric of interest is completion time. This is defined as the time from the first link being activated to the last link being deactivated. For example, the completion time of schedule b) in in Figure 1.3 is 2.5, which is 1.5 shorter as compared to schedule a).

This thesis aims to investigate the following hypotheses and research questions:

1. Past works have used affectedness \cite{29} to determine whether links in a set are able to meet their respective SINR requirement. A research question here is
whether *affectedness* can be used to lower completion time. Specifically, how *affectedness* can be used to construct transmission sets whereby links have a different activation time and data rate. Each transmission set may not include all links that can be active concurrently. Hence, the interference between links is less and all links use different data rates based on their SINR. Consequently, the completion time may be reduced further.

2. Machine learning is now gaining significant interest from both researchers and practitioners [30]. A fundamental question here is whether machine learning techniques can be used to schedule links. In particular, can nodes use methods such as reinforcement learning [31] to determine the optimal data rate under varying channel gains? In particular, how do nodes use reinforcement learning to interact with the environment, and use received feedback or reward to schedule links? Another hypothesis is that nodes are able to use reinforcement learning in a distributed manner to learn a transmission schedule. This is significant because a central node is not required to derive a schedule for all nodes. Moreover, nodes can adapt to random channel gains or changes in network topology locally.

3. As shown in [16], one method to achieve IBFD is via MIMO technology. One approach to allocate antenna elements is via the Degree of Freedom (DoF) model [32]. Specifically, the DOF of a node corresponds to the number of antennas it is equipped with. Nodes are able to cancel a number of interfering streams by consuming an equal number of DoF. A key problem that arises
is DoF or antenna allocation. That is, an algorithm is required to allocate sufficient antennas to cancel interfering streams and the self-interference in full-duplex transmissions, while maximizing the number of antennas used for data streams. A key research question here is how to use the DoF model when allocating IBFD links. In particular, how the antenna elements or DoFs of nodes are to be allocated to minimize the schedule length.

4. Nodes have varying traffic arrival rates. An important issue is to ensure the queue of nodes remain short or that they do not experience any buffer overflow. To this end, a key research question is how nodes are able to learn in a centralized manner to minimize packets drops when packet arrivals are random and their queue state is unknown.

To answer the above questions, this thesis considers three different network models. Briefly, the first model considers optimizing completion time of a given set of packets in a dense Wireless Local Area Network (WLAN). The second model considers an ad-hoc collection of nodes, where the goal is to learn a transmission schedule in a distributed manner. The last model corresponds to a WLAN cell or Basic Service Set (BSS) with nodes that have MIMO capability that experience exogenous interference and have random packet arrivals.

### 1.1.1 Minimizing Completion Time

As mentioned, completion time is a key metric to be optimized by a link schedule. The challenge, however, is the interference between links, meaning if links interfere with one another, they may have to be scheduled at a different time. Alternatively, they may lower their data rate, which allow links to co-exist with one another but at the expense of a longer transmission time.

To illustrate the problem, consider the dense WLAN shown in Figure 1.4. The two APs, labeled as 1 and 5, have six clients. Figure 1.5 shows example schedules for the links in Figure 1.4. If IBFD is not supported, the only available schedule
1.1. Problem Space and Motivation

will be schedule-a) where all links are activated one by one. It also has the longest completion time. If IBFD is supported, both schedule-b) and c) can be used. In schedule b), the number of links in each time slot is maximized because link 2 to 1 and link 1 to 2 cannot co-exist with link 1 to 3. Link 7 to 5 and link 5 to 7 cannot co-exist with link 6 to 5. Thus, to maximize the number of links in each time slot, the first slot contains four links and forms two bi-directional transmissions. The second slot then contains the remaining two links. However, the completion time is not the shortest because node-3 is interfered by node-6. Hence, the problem at hand is how to balance the number of links in each slot and the data rate for each link. In schedule c), link 1 to 3 and link 6 to 5 are activated in two slots. Although each slot only has one link, node-3 does not suffer interference from node-6. Consequently, both link transmit at a higher data rate and the completion time is the shortest.

Figure 1.4: A dense WLAN with a central controller, and all devices operate on the same frequency. Also shown are full-duplex links, as indicated by double headed arrows.

1.1.2 Varying Channel Condition

To address the second and third hypotheses, this thesis develops a distributed link scheduler for a wireless network where nodes have an IBFD radio. Unlike prior schedulers that only consider a predetermined link schedule with fixed or estimated channel gains, nodes are able to learn and select the best action through reinforce-
1.1. Problem Space and Motivation

Consider the mesh WLAN shown in Figure 1.6. Assume all nodes have packets and want to transmit at the current time slot. However, if more than two nodes transmit at the same time, all transmissions will fail. In existing works, all nodes have to contend for the opportunity to transmit, or all nodes are assigned with a specific time to transmit. In this thesis, nodes decide by themselves whether to transmit. An important challenge is to for nodes to learn whether their transmission will interfere severely with a neighbor’s transmission. In this regard, an important issue is to determine the current state of the system. Another consideration is random channel gains. Consequently, prior works that assume block fading are no longer applicable. Therefore, the last problem is how to determine the data rate without any knowledge of channel gains.

1.1.3 Random Packet Arrivals and Interference

Consider the example shown in Figure 1.7 where one AP is associated with two clients $a$ and $b$. Both the AP and clients are MIMO and IBFD capable. Assume all three nodes are equipped with five antennas and are only able to store two packets. The AP has one data packet for each client. Client-$a$ has two data packets for the AP and client-$b$ has one data packet for the AP. Client-$a$ is suffering from

![Figure 1.5: Example schedules.](image-url)
three interfering streams from a neighboring cell and client-\textit{b} is experiencing four interfering streams. Using the DoF model [32], the AP can take one of the following actions: 1) the AP downloads one packet to client-\textit{b}, 2) the AP asks one client to upload, 3) the AP downloads to one client and asks another client to upload at the same time, or 4) the AP downloads to one client and the client uploads to the AP at the same time. If the AP knows how many packets the clients have and how many interfering streams they are experiencing from neighboring cells, it is easy for the AP to determine if asking client-\textit{a} to upload while downloading to client-\textit{b} is the best choice because there will be three data streams. For any other actions, the number of data streams will be less. In addition, if client-\textit{a} is not asked to upload, client-\textit{a} will start to drop packets. The AP is also able to allocate one antenna for download, two antennas for upload and another antenna to cancel the interference between download and upload streams. However, collecting information from all clients will be time-consuming when the number of clients is large. On the other hand, when the AP has no information from clients, it is difficult for AP to determine which one of available actions will result in most data streams. Therefore, the AP needs to decide on the best action without all information from clients.
1.2 Contributions

This thesis contributes the following link schedulers to the state-of-the-art.

1.2.1 Minimizing completion time in IBFD WLANs

First, it presents three novel link scheduling algorithms that aim to minimize completion time. The three algorithms are able to add a set of links at any time instead of on a slot-by-slot basis. This thesis algorithms also allow links to have different data rates and activation time. The three algorithms adopt for the first time the concept of ‘affectedness’ [29] for scheduling both half-duplex and full-duplex links and consider three types of interference: 1) self, 2) cross, and 3) exogenous. This thesis studies the impact of different node densities and transmission power levels on link schedules; both of which govern the interference experienced by nodes, and hence, their data rates or transmission times. This thesis also considers different SINR thresholds, which affect the data rate employed by a link given its SINR value. The results show the first algorithm has the second best average performance, with a reduction in completion time of around 40% as compared to having all links transmit individually. The second algorithm performs better than the first algorithm if the interference between links is strong. The third algorithm has the best average performance under all scenarios but incurs the longest computation time.
1.2.2 A Distributed Q-Learning Based Link Scheduler

This thesis proposes a reinforcement learning approach that not only helps nodes learn which links to activate but also the highest possible data rate for each activated link. In addition, the proposed scheduler is distributed, where nodes select actions that maximize the overall throughput without the help of a central entity/node. The scheduler allows nodes to set up full-duplex transmissions with the optimal data rate under varying channel condition, and reverts to half-duplex transmissions when path loss is high to ensure successful transmissions. To the best of the author’s knowledge, there is no other link scheduler that employs reinforcement learning [31] to schedule nodes with an IBFD radio. The simulation results show that when nodes use our approach, their average throughput is triple that of Carrier Sense Multiple Access (CSMA), and up to quadruple the average throughput of Time Division Multiple Access (TDMA). Moreover, our link scheduler remains superior when channel gains vary significantly from their average value.

1.2.3 Scheduling Packets over the DoF Model

This thesis outlines a centralized Q-learning based scheduling algorithm that is able to allocate half/full duplex links under the DoF model [32] while minimizing packets drops. The proposed scheduler in Chapter 5 is also able to allocate antenna resources to cancel random interfering streams from nearby cells. The AP is only required to poll up to two clients to collect information in each time slot, instead of collecting information from all clients. The simulation results show the centralized Q-learning based scheduling algorithm increases the average number of data streams by about 60% as compared to traditional polling methods which also has perfect knowledge of all clients. The proposed scheduler in Chapter 5 also reduces packets drops by about 15%. However, when nodes always have packets to transmit, the performance of the proposed scheduler converges to that of the random polling method.
1.3 Publications

The research carried out in this thesis has resulted in the following articles:


1.4 Thesis Structure

1. *Chapter 2*. This chapter contains a survey of legacy MACs, full-duplex MACs and MACs related to various technologies, including multi-channels, directional antenna array, MIMO, IBFD and machine learning.

2. *Chapter 3*. This chapter presents three heuristic link scheduling algorithms which aim to minimize the completion time and give a non-slot-based schedule.

3. *Chapter 4*. This chapter outlines a Q-Learning-based link scheduler that runs in a distributed manner to maintain high throughput under varying channel condition.

4. *Chapter 5*. This chapter proposes a Q-Learning-based link scheduling algorithm for centralized WLANs. The algorithm is run by the AP and aims to maximize the number of data streams in each slot. It also considers nodes with random number of interfering streams from other cells.
5. *Chapter 6.* This chapter concludes the thesis, presents a summary of key contributions and possible future research directions.
Chapter 2

Literature Review

A Medium Access Control (MAC) protocol is responsible for scheduling transmissions over a shared media [33]. In general, MAC protocols can be divided into two categories: contention and contention free. Contention-free schemes coordinate channel access in the time, frequency, or coding domain. Schemes that use Time Division Multiple Access (TDMA) [34] rely on a central station to assign each device a fixed time slot. Each device is only allowed to transmit in its assigned slot. To maximize spatial reuse, slots can be assigned using two scheduling strategies: node-oriented [35] or link-oriented [36]. Node-oriented strategies assign transmission on a node-level. Concurrently transmitting nodes are assumed to be two hops away from each other to avoid interference. A node can transmit to any intended receiver(s) in its assigned time slot. On the other hand, link-oriented strategies assign links to time slots. Link oriented strategies have higher spatial reuse [36]. For Frequency Division Multiple Access (FDMA) based MACs, each device is assigned an orthogonal channel. In Code Division Multiple Access (CDMA) [37], each client is assigned an orthogonal code. Thus, multiple transmissions can co-exist in the same time-frequency space. To achieve the same bit rate as other contention-free schemes, CDMA requires a higher bandwidth. The final category of contention free schemes is polling [38]. It requires a master node to send a polling signal to each
2.1 Physical Layer Technologies

A high spatial reuse or concurrent number of links translates to a high network capacity. A key challenge to achieving high spatial reuse is managing interference. To date, researchers have proposed many strategies to minimize interference. These strategies include:

1. **Multiple channels**: Multiple channels [43] improve performance because they allow links to operate on orthogonal channels. They bring many advantages. Firstly, they enable conflict-free transmissions as long as nodes are tuned to a non-conflicting channel [3]. Thus, more simultaneous data transmissions are possible. In fact, the latest WLAN technology, namely IEEE 802.11ax [44], uses Orthogonal Frequency Division Multi Access (OFDMA) to allow simultaneous data transmissions by dividing the frequency spectrum.
into narrow-band sub-carriers [45]. Secondly, the use of multiple channels helps lower channel contention time. A node can use a channel to exchange control packets to reserve a channel for its data frames. An example MAC is by Li et al. [4], where RTS/CTS exchange is executed over a dedicated control channel and data is transmitted over a different channel. Thirdly, a channel can also be used to send a busy tone in order to avoid hidden terminals. For instance, the Dual Busy Tone Multi Access (DBTMA) scheme [6] uses a transmit-busy and receive-busy tone to protect not only data transmissions but also control packet exchanges.

2. **Directional or adaptive array antennas**: Unlike an omni-directional antenna, which radiates power uniformly in all directions in one plane, a directional antenna or adaptive array radiates power much greater in one specific direction than other directions [8]. Hence, directional antennas are able to increase coverage and connectivity because of their higher antenna gains. In addition, it provides higher spatial reuse because it radiates at a specific geographical area and thus minimizing interference to other areas [8]. This fact has a direct impact on network capacity. Yi et al. [46] show that network capacity can be improved by a gain of \( \frac{2\pi}{\sqrt{\alpha\beta}} \) for an arbitrary network and by a gain of \( \frac{4\pi^2}{\alpha\beta} \) for a random network, where \( \alpha \) is the main beamwidth of a transmitting antenna and \( \beta \) is the main beamwidth of a receiving antenna.

3. **Multiple Input Multiple Output (MIMO)** [14]. Equipping nodes with several antenna elements provide two types of advantages: spatial diversity gain and spatial multiplexing gain. The former refers to the ability to transmit the same data or symbol out on multiple antenna elements. This improves the Signal-to-Noise Ratio (SNR) at a receiver, and thus MIMO technology helps overcome severe fading and improves reliability. Spatial multiplexing gain refers to the ability to transmit data on different spatial channels. This leads to two different MIMO schemes: single and multi-user MIMO. Single-user MIMO
2.1. Physical Layer Technologies

focuses transmissions on a single destination. This advantages as the system capacity increases linearly with the minimum number of antennas between the transmitter or receiver [15]. Multi-user MIMO (MU-MIMO) allows a node to communicate with multiple destinations. In this case, the system capacity improves by \( \min(aN_t, bN_r) \), where \( a \) is the number of sources, \( b \) is the number of destinations, \( N_t \) is number of transmitting antennas, \( N_r \) is the number of receiving antennas [15].

The key challenge when employing MIMO is to acquire the Channel State Information (CSI) [14]. For example, an Access Point (AP) can estimate the CSI to stations via a training sequence sent by stations; aka implicit feedback. Conversely, the AP can use explicit feedback, whereby stations first measure the CSI based on the training sequence sent by the AP. They then send the measured CSI to the AP. Details of these two schemes can be found in references [47] and [48]. The CSI of a MIMO link with \( N_t \) transmit antennas and \( N_r \) receive antennas can be presented in an \( N_t \times N_r \) matrix. Each element, denoted as \( h_{tr} \) in the matrix presents the channel gain between transmit antenna \( t \) and receive antenna \( r \) [49–53]. Alternatively, the Degree of Freedom (DoF) model offers a significantly simpler representation as compared to the traditional matrix-based representation [32]. The DoF model only requires simple additions and subtractions to track spatial multiplexing and interference cancellation [54]. Consequently, sufficient condition for feasible data streams is easier to be identified under the DoF model. The details of DoF model are outlined in [17, 32, 54–58]. This thesis will further outline the rules for DoF allocation in Chapter 5.

4. **Successive Interference Cancellation (SIC):** This technology enables multiple packet reception. A receiver using SIC first decodes the strongest signal, subtracts it from the received signal and repeats the process to recover other transmissions [59]. There are various methods to achieve SIC, including
MIMO, CDMA and OFDMA.

5. **In-Band Full Duplex (IBFD)**: A node equipped with an IBFD radio is able to transmit and receive data concurrently over the *same* frequency [26]. IBFD has been shown to achieve a 1.47x gain as compared with half-duplex in an ad hoc network under the utility-optimal CSMA scheme [27]. IBFD enables two new transmission scenarios: bi-directional and relay, see Figure 1.2. A key challenge in enabling IBFD is how to cancel self-interference [26]. The goal is to reduce the self-interference power from a node’s own transmission antenna chain to its receive antenna chain. Research on self-interference cancellation can be categorized into the *passive* and *active* scheme.

An example of a passive or propagation-domain scheme is to increase the distance between the transmit and receive antenna. Another way is by placing the receive antenna at a point $d$, and a second antenna at point $d + \frac{2}{\lambda}$ and the transmit antenna at point $d + \lambda$, where $\lambda$ is the wavelength of the carrier. The self-interference power will be weakened due to the signal from the second antenna and the transmit antenna adding destructively [24]. However, both methods are limited by the space on devices. The second method is also limited by the carrier frequency. Another passive scheme is to exploit different polarization. For example, a receive antenna is tuned to receive horizontally polarized signals while the transmit antenna uses vertical polarization. Polarization is limited in MIMO systems. Therefore, advanced methods are used to improve electromagnetic isolation, such as using a band-gap structure to prevent surface waves [60], using inductive loops to generate counter-flowing magnetic fields [61] and using ground plane slots to reduce coupling [62].

Active schemes can be divided into analog and digital schemes. Subtracting an estimated self-interference signal is the major approach for analog cancellation, which aims to suppress self-interference before it enters the Analog-Digital Converter (ADC). Existing technologies can either be non-adaptive or
adaptive. Non-adaptive methods use fixed parameters and may require manual configuration. For example, the noise cancellation chip QHx220 which requires manual configuration of amplitude and phase of interference reference signal [24].

As for adaptive methods, circuit parameters are adjusted automatically according to the reflected signal from transmitting antennas. For example, Balun cancellation [63] allows a Balun-based circuitry to automatically adapt and cancel self-interference. This circuit has been experimentally reported to suppress self-interference by up to 72 dB [64]. Digital cancellation also aims to subtract the estimated transmit signal from the receive signal. To estimate the transmitted signal, it requires methods such as minimum mean square error filters, zero-forcing beam-forming, and null space projection [65, 66].

The following subsections briefly review MACs that take advantage of the aforementioned technologies to maximize network capacity or minimize interference, except IBFD. As this thesis focuses on IBFD capable nodes, Section-2.2 will provide a detailed review of MACs that take advantage of bi-directional full-duplex or relay communications.

### 2.1.1 Multiple Channels

Multiple channel MACs face a new challenge. In particular, they need a channel assignment strategy [43]. Currently, there are three major assignment strategies: reservation, signaling, and hybrid [43]. A number of works [5–7, 67–71] have studied various channel assignment strategies. For reservation strategies, they schedule data transmissions over one or more dedicated control channel(s). For example, nodes exchange RTS/CTS messages over a separate control channel in [67]. The RTS frame piggybacks a list of channels used by a transmitter and the CTS frame piggybacks a list of free channels at the receiver. Then, the transmitter sends a reservation packet to inform the receiver and neighbouring nodes its selected chan-
2.1. Physical Layer Technologies

Based on [67], the transmitter in [69] will always choose the channel that has the best channel condition and uses the minimum transmission power. The MAC in [68] is also based on [67]. Instead of letting the transmitter choose the channel via a reservation packet, the receiver in [68] informs the transmitter the channel with the best condition via a CTS frame. Dual Channel Pipelined Scheduling (DCPS) [5] separates the RTS/CTS and Data/ACK exchange over two channels. Nodes carry out the RTS/CTS exchange for the next transmission during an on-going data transmission. For signaling strategies, the dedicated control channel is used for signalling instead of control frame exchanges. For example, in [70], a base station broadcasts a busy tone over a control channel to prevent nearby nodes from transmitting when it senses the data channel is busy. Nodes that use Dual Busy Tone Multiple Access in [6] use two different types of busy tones to indicate whether they are transmitting or receiving. Hybrid strategies use signalling and control frames together [7, 71]. In the Dual-Channel MAC [7], each node has three non-conflicting channels. It conducts RTS/CTS exchange on one channel and broadcasts a busy tone on another channel during data transmission, which is conducted over a third channel. This means an RTS/CTS exchange can be successful even when exposed nodes are transmitting data. The nodes in [71] sense the busy tone twice, before and after an RTS/CTS exchange to find out which node is sending the busy tone.

2.1.2 MACs with Directional or Adaptive Array Antennas

The use of directional antennas results in a new hidden terminal problem. In particular, the traditional RTS/CTS exchange will fail as these control packets only can be heard by some nodes as opposed to all nodes [8]. To solve the hidden terminal problem, the IEEE 802.11-DCF protocol has to be modified. For example, in [72], RTS/CTS frame is sent omni-directionally to solve the hidden terminal problem. Only data is sent directionally. In [10], the RTS frame, ACK frame and data are sent directionally. The CTS frame is sent omni-directionally to avoid collisions. The
MAC in [73] uses the the RTS/CTS exchange of [10]. However, the control frame also contains information about two types of neighboring nodes: direction-omni and direction-direction. If one node listens to the channel omni-directionally but can receive a directional transmission from another node, then these two nodes are direction-omni neighbours. Two nodes are directional-directional neighbours if they are only able to receive directional transmissions from each other when they listen to channel directionally. Based on neighbouring nodes information, a transmitter is able to determine a route for its intended receiver which is multiple hops away. In [74], nodes always listen to the channel and record the location information of nodes whenever they overhear signals. If a node has the location information of an intended receiver, it sends the RTS frame directionally. Otherwise, the RTS frame is sent omni-directionally. If the node does not receive the CTS frame, it re-sends the RTS frame omni-directionally. In [75], RTS/CTS messages are sent directionally. All idle nodes always listen to the channel. If a node overhears an RTS/CTS frame from one direction, it only defers its own transmission in that direction. Each node also broadcasts a unique busy tone omni-directionally after each successful transmission. Neighbouring nodes that overhear the busy tone will reduce their contention window to the minimum size. In [76], RTS/CTS frames, data and ACK are sent directionally. While exchanging data, a busy tone is sent omni-directionally to avoid collision.

Non-802.11 based solutions also exist. One example is the Direction-Of-Arrival MAC (DOA-MAC) [9]. It is based on Slotted-Aloha but further divides the time slot into three mini-slots. In the first mini-slot, each transmitter sends a tone signal and each receiver tunes the direction of their main beam by running a direction-of-arrival algorithm. Then the second mini-slot is used for data transmission. The third mini-slot is used for acknowledgment. In [77], nodes are equipped with a multi-beam adaptive antenna array which can form multiple beams for multiple transmissions or receptions. The proposed MAC in [77] assigns each node with a priority number based on the IDs of nodes and the current time slot. If the priority number is
2.1. Physical Layer Technologies

odd, a node transmits. Otherwise, a node enters the receive mode. The node that transmits will select $K$ neighbouring as receivers based on their priority number, where $K$ is the number of beams that can be formed simultaneously.

2.1.3 MIMO MACs

The key challenge when designing MIMO MACs is how to schedule uplink or downlink communications. In existing schemes, an AP plays an important role. For example, Cai et al. [20] propose a distributed MU-MIMO MAC downlink protocol. RTS and CTS messages are extended to include CSI. The AP schedules transmissions based on CSI and the buffer state at the AP, whereby packets are scheduled according to their queued time.

As for uplinks, scheduling becomes more difficult because clients are distributed and thus making it harder to obtain CSI and queue information to schedule links. Schemes to schedule uplinks can be divided into coordinated and uncoordinated. In the former, an AP extracts information from RTS packets sent by contending clients. Then it decides which clients can transmit based on different strategies [14]. As an example, in [19], the AP replies with a pilot-requesting CTS after it receives an applying-RTS. Then STAs transmit a sequential pilot to the AP for channel estimation. Finally, the AP sends a notifying-CTS message to the selected clients that have a good channel condition. Clients that received a notifying-CTS message start their transmission. In uncoordinated schemes, the AP no longer takes part in contention. Clients utilize a random MAC. These schemes can be further divided into synchronous and asynchronous. A typical example of synchronous schemes is proposed by Jin et al. [22]. In [22], after the channel is idle for a IEEE 802.11 DCF inter-frame space period, all uplink transmissions start simultaneously. After the transmissions are finished, the AP sends ACK one by one to each client that has transmitted. The clients do not start to count down for ACK timeout as long as they can overhear any ACK frame being transmitted. Asynchronous schemes
do not require clients to start transmission simultaneously. Their transmission can start at any time before the next round of contention. For example, Tan et al. [18] propose an asynchronous protocol. In their so called Spatial Multiple Access (SMA) scheme, an AP will firstly broadcast in its beacon a threshold that specifies the maximum number of transmissions. The first client that wins contention will start its transmission right after its back-off expires. Other clients will back-off to another random period. If they sense the number of concurrent transmissions have not reached a threshold, they will start their transmission. The process will repeat until the number of concurrent transmissions reaches the threshold advertised by the AP.

There are also many works that have applied the DoF model [32, 55–58]. In [78], the authors show how a transmitter adjusts its antenna to nullify its interference to unintended receivers, and how a receiver adjusts its antenna to eliminate interference from unwanted transmitters. In [17], the authors propose a linear optimization algorithm to maximum the throughput for MIMO networks under the DoF model. In [79], the authors propose a centralized and a distributed stream MAC protocols. In both protocols, the links included in multiple contention domains are ranked as red links. Those in a single contention domain are ranked as white links. Given these links, both protocols allocate resources to red links and white links following the rules of DoF allocation. In [80], the authors propose an algorithm that schedules links over multiple time slots. In [80], when multiple links have to be assigned into the same time slot, the proposed algorithm allocates antenna resources based on the DoF model to avoid interference, or it allocates different bandwidth to each link. In [81], the authors propose a distributed scheduling algorithm for ad hoc networks where nodes are equipped with a cognitive and MIMO radio. The algorithm has two modules. The first module is for channel assignment. The second module is for stream allocation based on the DoF model when multiple links are assigned with the same channel. The authors in [57] propose a greedy coloring algorithm that aims to maximize the number of data streams and match a known traffic demand. In [54],
the authors aim to maximize the throughput of a multi-hop MIMO network. They propose an iterative greedy algorithm with three modules. The first module orders the links based on their potential interference towards other links, and sequentially adds one link into the schedule every time. Then, the second module allocates antenna resources. Lastly, the third module re-orders the links in the schedule and tries to relieve DoF resources from interference cancellation to data streams.

2.1.4 SIC Aware MACs

The key challenge with SIC is its physical implementations; see [82] for details. In summary, SIC is achievable through MIMO, OFDMA and CDMA. Hence, the MACs in Section 2.1.3 are all also SIC aware. Hence, this section will highlight works that focus on SIC at layer-2 of the protocol stack rather than physical layer aspects of SIC.

In [83], the authors propose a heuristic scheduling algorithm. The algorithm adds one link at a time into existing schedule as long it does not cause excessive interference to other links. In each iteration, the algorithm selects the link that causes the minimum interference to scheduled links. The authors in [84] propose a simultaneity graph to capture the effect of successive interference cancellation. Then, they present an independent set based greedy algorithm which gives a schedule with the maximum number of links. In [85], nodes equipped with an IEEE 802.11 based MAC along with SIC allows neighbouring nodes to start a second transmission along with an on-going transmission. Neighbour nodes will send a channel-condition-request packet with a certain activation probability after an RTS/CTS exchange between a transmitter and a receiver. Then, after receiving a channel-condition-request packet, the transmitter and receiver will use SIC decoding to determine whether their transmission will be interfered when neighbouring nodes start transmission. If not, they remain silent and neighbour nodes start their transmission. Otherwise, they broadcast a busy tone, which causes neighbouring nodes to back-off. In [21], the authors
2.2 Full-Duplex MACs

Full-duplex MACs aim to exploit full-duplex opportunities whenever possible [26]. A simple approach is shown in [63], where a client simply starts to transmit to its associated AP whilst receiving if it has packets. If the client finishes first, then it broadcasts a busy tone until it finished receiving. The drawback of this simple approach is that it does not support relay transmissions. To fully explore full-duplex opportunities, one approach is to modify the IEEE 802.11 DCF. For example, a novel full-duplex MAC is proposed by [86]. It replaces the CTS packet of CSMA/CA with a Full Duplex Clear-to-Send (FCTS) packet. The FCTS packet contains the address of the secondary receiver and the duration of the secondary transmission. It also includes the node that is ready for full-duplex transmission. The corresponding RTS packet contains the primary receiver and the duration of the primary transmission. A full duplex transmission starts with the standard CSMA/CA. The node that wins contention becomes the primary transmitter. The primary transmitter sends an RTS packet to the primary receiver. This leads to two scenarios, depending on the queue state of the primary receiver. Namely, 1) if the primary receiver has packets for the primary transmitter, it sends an FCTS packet to the primary transmitter. Then, bi-directional transmission starts, 2) if the primary receiver has packets for another node, it sends an FCTS packet to the primary transmitter and a third node. Then, if the third node is idle, the third node that is now the secondary receiver sends
back a FCTS packet to the primary receiver. Finally, relay transmission starts.

In [87], the proposed MAC enables bi-directional and relay transmission using a FCTS packet [86]. Control frames are transmitted at the maximum power to warn hidden terminals. After exchanging control frames, data frames are transmitted using a transmit power that alternates between a given minimum and maximum values. Another IEEE 802.11 DCF based full duplex MAC protocol is proposed in [88]. The proposed MAC considers two modes: 1) destination-based relay, in which the secondary receiver is selected by the primary receiver, 2) source-based relay, in which the primary transmitter receives from one of its neighbouring nodes excluding the primary receiver. The proposed MAC introduces a new field to represent the intended transmission mode. The channel reservation procedures for half-duplex, bi-directional and destination-based relay transmissions are similar to [86]. A half-duplex transmission is reserved using traditional RTS/CTS handshaking. Bi-directional or destination-based relay transmissions are carried out using RTS/CTS/CTS three-way handshake. As for the source-based relay mode, the primary transmitter and the primary receiver start their half-duplex transmission first. Neighbouring nodes of the primary transmitter then start sub-carrier contention, where they randomly select a sub-carrier to broadcast a busy tone. Neighbouring nodes that choose the same sub-carrier as the primary transmitter win the sub-carrier, and start to transmit to the primary transmitter.

The work in [89] introduces a MAC that uses a shared random backoff. To start a full-duplex transmission, either an AP or a client needs to start a half-duplex transmission using CSMA/CA. If the receiver has a packet for the transmitter, it will inform the transmitter through an ACK packet. The transmitter then evaluates whether it has packets for the receiver. If so, it sends another ACK to the receiver to indicate the duration of the full-duplex transmission and a shared backoff value. Both the transmitter and receiver commit to a shared random back-off before they start full duplex transmission. Other clients also contend for the channel at the same time. If a third client wins the channel, the transmitter and the receiver give up their
2.2. Full-Duplex MACs

bi-directional full duplex transmission. The third client will initiate a half-duplex transmission.

The Semi-MAC in [90] assumes all transmissions are bi-directional. It introduces semi-synchronous channel contention. First, all nodes access the channel using CSMA. The node that wins contention becomes the primary transmitter and starts transmission immediately. If the primary receiver hears the transmission from the primary transmitter, and if it has any packets, it also starts transmission immediately. Otherwise, the primary receiver sends a busy tone. If the transmission from the primary transmitter is unsuccessful or the primary receiver is receiving from another node, the primary receiver remains silent. This means that the primary transmitter will hear nothing from the primary receiver. After some time, the primary transmitter notices the transmission has failed. It will then contend for re-transmission in the next round.

Relay-Full-Duplex MAC (RFD MAC) [91], Rapid concurrent transmission coordination MAC (RCTC MAC) [92] and Contraflow MAC [93] aim to reduce collisions in distributed networks. RFD MAC [91] focuses on reducing collision between primary and secondary transmissions. As shown in Figure 2.1 (a), the transmissions on link (A, B) and link (C, D) belong to different flows. If they are enabled simultaneously, there will be a collision at both node B and D. The interference power can be high because node D can be close to node A. Another scenario is shown in Figure 2.1 (b), the transmissions over the link (A, B) and (B, E) belong to the same flow. If they are enabled simultaneously, then the collision will happen at node E. The interference power also can be low because node E and node A are out of communication range from each other. Otherwise, node A is able to communicate with node E directly. Hence, there are fewer collisions as compared with the scenario shown in Figure 2.1 (a). To ensure both transmissions belong to the same flow, nodes listen to the transmissions of neighboring nodes. They then construct a table that records whether a neighbor has packets and their destination. After a node wins the CSMA/CA contention, it becomes a primary transmitter and selects
2.2. Full-Duplex MACs

Figure 2.1: Collisions between flows in full-duplex transmissions.

A secondary transmitter based on the information of the surrounding node table. The secondary transmitter can only be on either the previous-hop or next-hop of a given flow.

In RCTC MAC [92], control frames are replaced by pseudo-random noise sequences. Each node has a unique ID within one hop. Each node can generate pseudo-random noise sequences of two different types: 1) pseudo-random noise sequence that are based on the IDs of nodes, 2) a control pseudo-random noise sequence that represents the preferred transmission mode. It uses a similar procedure as in [86] to start a transmission. However, RCTC MAC exchanges pseudo-random noise sequences instead of control frames. To reduce collision, every node maintains three tables: ExMap, SecMap, and $\alpha$. The ExMap table records the probability of a successful transmission to different nodes when the node is an exposed node. The SecMap table records the probability of a successful transmission to a secondary receiver when the node is a secondary transmitter. The $\alpha$ table records the successful transmission probability when the node is a primary transmitter. When the node is an exposed node, it has a higher probability of starting a new transmission if its ExMap indicates that a potential receiver has a highly successful transmission possibility. When a node is a secondary transmitter, it always selects the receiver that has the highest successful transmission possibility. If the primary receiver has a successful transmission probability that is lower than a threshold, the primary transmitter broadcasts a special pseudo-random noise sequence before it initializes any
transmission. This special pseudo-random noise sequence informs all exposed nodes to remain silent during the transmission initialized by the primary transmitter.

In Contraflow MAC [93], the node that wins contention becomes the primary transmitter and sends its MAC header to the primary receiver. As soon as the primary receiver successfully decodes the MAC header from the primary transmitter, the primary receiver chooses a secondary receiver and sends another MAC header. Then, both the primary transmitter and primary receiver start transmission. If a transmission completes earlier than the other one, a busy tone is transmitted by the node that finishes first. Each node also maintains a list containing the address of nearby nodes and a weight that indicates their probability of receiving packets successfully. A high weight value means a high probability of success. When a node chooses a receiver, it only chooses the node that has a high weight value. Furthermore, in terms of fairness, each node holds a pressure indicator that increases in value if the node does not transmit in a time slot or its transmission fails. Otherwise, it decreases. A high-value pressure indicator means the node has a higher probability of winning access to the channel or become a secondary transmitter.

The full duplex MAC in [94] aims to eliminate the use of a busy tone for bi-directional transmissions with an asymmetric transmission time. In this protocol, a bi-directional transmission starts with IEEE 802.11 DCF. The node that wins contention becomes the primary transmitter. The authors assume that the primary receiver always has data for the primary transmitter. Therefore, the primary receiver replies with a Full-Duplex Clear-to-send (FD CTS) packet. Then, bi-directional transmission starts. The transmission is considered to be asymmetric where the primary transmitter always finishes transmission earlier. The primary receiver will pause the data transmission towards the primary transmitter and sends an ACK. The primary transmitter then sends a flagged packet. The neighbor nodes around the primary transmitter that hear the flagged packet will continue to freeze their back off counter until the time indicated in the flagged packet. The neighbor nodes around the primary receiver that hear the ACK packet will start IEEE 802.11 DCF
contention under the following scenarios: 1) they have data for the primary receiver, 
2) they can finish the transmission towards the primary receiver before the primary 
receiver finishes transmission towards the primary transmitter.

The full duplex MACs discussed thus far assume nodes have an omni-directional 
antenna. However, self-interference can be reduced through directional antennas. 
In order to exploit the benefits of directional antennas, Sugiyama et al. [95] pro-
pose a directional asynchronous full-duplex medium access control for distributed 
networks. All nodes have full-duplex capability and directional antennas. The node 
that wins the channel becomes the primary transmitter. It sends an RTS packet 
to a primary receiver, which then sends a Ready and Clear to Send (RCTS) packet 
to both the primary transmitter and a secondary receiver. This packet confirms 
the transmission from the primary transmitter and also requests the transmission 
to the secondary receiver. Then, the primary transmitter sends a Set Network Al-
location Vector (SNAV) packet to neighbor nodes to defer their transmission. The 
primary transmitter then starts the primary transmission to the primary receiver. 
At the same time, the secondary receiver broadcasts a Destination Set Network Al-
location Vector (DSNAV) packet in the opposite direction of the primary receiver. 
The DSNAV packet informs neighbor nodes around the secondary receiver to de-
fer their transmission. There is no collision because the receiving antenna of the 
primary receiver is pointed in the direction of the primary transmitter. Then, the 
primary receiver starts the secondary transmission to the secondary receiver. When 
transmitting data, the primary transmitter and the primary receiver choose two 
different directions using angle of arrival localization and global positioning system 
data. These directions ensure the primary transmission and the secondary transmis-
sion can reach their targetS. The two directions also ensure the two signals do not 
overlap with each other. As the secondary receiver cannot hear from the primary 
transmitter, the secondary receiver assumes a collision has happened.

As for centralized full duplex MACs, in [96], the authors assume the traffic load 
is asymmetric. Specifically, uplink traffic is always lower than downlink traffic.
The AP uses IEEE 802.11 DCF for downlink traffic in every transmission round. Clients use two channel access methods: 1) if their queue has reached a certain threshold, they use IEEE 802.11 DCF, and 2) if the AP is communicating with a client and there is no uplink, the client starts uplink transmission if it has data. If the uplink transmission completes first, the client will transmit a busy tone until the downlink completes. The MACs in [97–99] are also based on IEEE 802.11 DCF but not limited to asymmetric traffic. They aim to improve the overall throughput by giving priority to nodes with a better channel or more traffic demand during IEEE 802.11 DCF contention. The MAC protocol in [98] is based on [96]. However, the AP has two different contention windows: small and large. The AP chooses one of the contention windows dynamically according to traffic demands. When the demand for downlinks is large, the AP uses a small contention window where downlink has priority to occupy the channel. On the other hand, when the demand for uplinks is large, the AP uses the large contention window and thus giving clients a higher priority to transmit. Power-Controlled MAC [99] introduces the Received-Signal-Strength-Based (RSSB) contention scheme. The authors assume the receiving gain of each client is known and the interference between downlink and uplink at each client is also known. At the beginning of each time epoch, the RSSB scheme adjusts the contention window of each client according to its channel condition. In particular, a client that has a better channel condition has a higher priority to occupy the channel. Upon completing random back off, a client sends an RTS frame to the AP. The AP then sends a CTS-Uplink frame as a response. Then, if the AP also has packets to download, the AP sends a CTS-Downlink frame to inform the intended receiving client. Finally, both the uplink and downlink start.

In [97], the AP calculates a channel access probability $p$ for each client at the beginning of each new time slot based on the clients historical traffic demands. The AP then chooses a sub-carrier randomly from multiple OFDM sub-carriers. The AP that chooses the sub-carrier with the smallest channel number wins access to the downlink channel. Then, this AP determines the transmission mode based on the
2.2. Full-Duplex MACs

channel access probability $p$ of each client. If full duplex clients have a high channel access probability, an AP chooses the full duplex mode. Similarly, if half duplex clients have a high access probability, the AP chooses the half-duplex mode. Then, clients run the standard CSMA/CA contention for uplinks. To ensure clients with a high access possibility win contention, those with a high access possibility use a small contention window. Lastly, to prevent starvation, if a client has an access possibility that is low, the AP replaces it with a default value.

MASTaR [100] is a full duplex MAC for indoor IEEE 802.11h Wi-Fi networks. A full duplex transmission always starts with an uplink transmission from a client. The client has two choices when the channel is free: 1) the client sends a data packet if the length of the data packet is less than a given threshold, or 2) the client starts an RTS-CTS exchange if the packet length is larger than a threshold. When the AP is receiving from a client, there are two cases: a) if the AP does not have data to send, it unicasts or broadcasts a transmit power control request frame to other clients. After the AP sends an ACK packet to the primary client, other clients that hear the transmit power control request frame send a report to the AP. The AP uses the report from clients to build an interference map to choose a client for downlink transmission, or b) if the AP has data packets for other clients, the AP will select a client that has less interference. Then, the AP sends a dummy packet to test the downlink channel. If the destination client does not indicate a reception failure, the AP sends the data packet that has a suitable length to ensure the downlink transmission finishes earlier than the uplink transmission.

Janus [101] is a non-IEEE 802.11 DCF based MAC. A centralized AP takes full control of FD transmissions. Janus aims to 1) identify full-duplex opportunities, 2) schedule transmissions, and 3) provide fairness. For 1), the AP firstly queries all registered clients to collect their traffic demand. For 2), the AP constructs an SINR matrix that represents the SINR of every downlink and uplink. This SINR matrix also reflects possible interference between links. Then, a rate-time allocator algorithm chooses active links. It randomly chooses a link and tries to pair it with
2.2. Full-Duplex MACs

A link that causes the minimum interference to all existing active links. Thus, all active links can transmit at a high data rate. The overall transmission completion time can be optimized. For 3), Janus uses uses the deficit round robin scheduler [102] to ensure each client receives a fair share of the channel capacity.

<table>
<thead>
<tr>
<th>Protocols</th>
<th>Contention Method</th>
<th>Self-IC</th>
<th>Traffic</th>
<th>Full-Duplex Modes</th>
<th>Topology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jain et al. [63]</td>
<td>Random access</td>
<td>Perfect</td>
<td>Synchronous</td>
<td>Bi-directional</td>
<td>Centralized</td>
</tr>
<tr>
<td>Goyal et al. [88]</td>
<td>RTS/CTS handshaking</td>
<td>Perfect</td>
<td>Synchronous</td>
<td>Bi-directional and relay.</td>
<td>Distributed</td>
</tr>
<tr>
<td>Cheng et al. [86]</td>
<td>RTS/CTS handshaking</td>
<td>Perfect/Imperfect</td>
<td>Asynchronous</td>
<td>Bi-directional and relay.</td>
<td>Distributed</td>
</tr>
<tr>
<td>Al-Kadri et al. [87]</td>
<td>RTS/CTS handshaking</td>
<td>Perfect/Imperfect</td>
<td>Asynchronous</td>
<td>Bi-directional and relay.</td>
<td>Distributed</td>
</tr>
<tr>
<td>Sahai et al. [89]</td>
<td>Random access</td>
<td>Perfect</td>
<td>Asynchronous</td>
<td>Bi-directional</td>
<td>Distributed</td>
</tr>
<tr>
<td>Xie et al. [90]</td>
<td>Random access</td>
<td>Perfect</td>
<td>Synchronous</td>
<td>Bi-directional</td>
<td>Distributed</td>
</tr>
<tr>
<td>Tamaki et al. [91]</td>
<td>RTS/CTS handshaking</td>
<td>Perfect</td>
<td>Synchronous</td>
<td>Relay</td>
<td>Multi-hop</td>
</tr>
<tr>
<td>Zhou et al. [92]</td>
<td>Random access</td>
<td>Perfect/Imperfect</td>
<td>Synchronous</td>
<td>Bi-directional and relay.</td>
<td>Distributed</td>
</tr>
<tr>
<td>Jain et al. [93]</td>
<td>Random access</td>
<td>Imperfect</td>
<td>Synchronous</td>
<td>Bi-directional and relay.</td>
<td>Distributed</td>
</tr>
<tr>
<td>Kim et al. [94]</td>
<td>Random access</td>
<td>Perfect</td>
<td>Synchronous</td>
<td>Bi-directional and relay.</td>
<td>Distributed</td>
</tr>
<tr>
<td>Sugiyama et al. [95]</td>
<td>RTS/CTS handshaking</td>
<td>Perfect</td>
<td>Asynchronous</td>
<td>Bi-directional and relay.</td>
<td>Multi-hop</td>
</tr>
<tr>
<td>Murad et al. [96]</td>
<td>Random access</td>
<td>Perfect</td>
<td>Synchronous</td>
<td>Bi-directional.</td>
<td>Centralized</td>
</tr>
<tr>
<td>Oashi et al. [98]</td>
<td>Random access</td>
<td>Perfect</td>
<td>Synchronous</td>
<td>Bi-directional.</td>
<td>Centralized</td>
</tr>
</tbody>
</table>
2.3 Learning Based MACs

Recently, machine learning techniques have been applied successfully to address various networking and communication problems [30]. For example, they have been used to improve routing [103], traffic classification [104], flow control [105] and link scheduling [106–109]. One popular machine learning technique is Reinforcement Learning (RL) [110, 111] in which an agent learns to execute the most rewarding action under each state/environment. The problem is usually modeled as Markov Decision Process (MDP) [30], which can be solved by value iteration [112] when the state transition probability is given, or Q-learning [113] when the state transition probability is difficult to obtain. More details of MDP and Q-learning are presented in Section 4.1.3. When an agent only has partial information of the state, then one can apply Partially Observable MDP (POMDP) [114]. In this section, existing works that relate to link scheduling or channel access and utilize RL and other machine learning techniques will be presented.

In [106], the authors aim to obtain the optimal link schedule for dense device-to-device networks using a neural network. The authors assume the channel condition is determined by the geographical location of the nodes. Their neural network is
trained in supervised fashion which captures the geographical location information of transmitters or receivers. Their scheduler yields the optimal schedule without requiring channel estimation. In [107], the authors model the packet transmission of a single time slot as a Markov Decision Process (MDP). They aim to improve the overall transmission reliability. The nodes learn to transmit packets over ‘good’ links, or delay transmissions when channel condition is poor. In [108], the authors implement a multi-agent cooperative reinforcement learning [31] algorithm for use over a two-hop network with energy harvesting nodes. These nodes learn to select different levels of transmission power to achieve the maximum throughput given their energy constraint. They also learn to operate cooperatively to exchange information about their incoming energy, concurrent channel condition, and current battery level. In [109], nodes use a deep-reinforcement learning algorithm to switch between conventional MAC protocols, such as Aloha or Time Division Multiple Access (TDMA), in each time slot to avoid collision and improve transmission success possibility. In [115], the authors develop a Q-Learning algorithm that requires multi-layers state information to allow nodes to determine their transmission probability at each time slot. The multi-layers state information includes how many slots in which the node’s queue is empty, number of collisions, and idle slots. In [116], the authors propose a learning automata based scheduling algorithm for wireless multi-hops networks. Each node uses a controller that runs two learning automata simultaneously. The first learning automaton lets a node learn whether it should participate in channel contention in a given slot. The second learning automaton helps a node improve its schedule over time. To determine the data rate of a link, the authors also introduce a conflict graph based on the physical interference model, and a distributed depth first search algorithm to split the overall conflict graph for each node. In [117], the authors model the scheduling problem as a combinatorial multi-armed bandits problem and propose a greedy algorithm. The greedy algorithm allows nodes to learn which link to activate in each slot; this is achieved using a conflict graph. A challenging issue is that the capacity of each link varies over time with a fixed
distribution. The authors in [118] propose a synchronized contention based algorithm, called Randomized Contention Aware Multiple Access (RCAMA). They use a frame containing multiple time slots. Links contend in each time slot. If a transmission over a link is successful, RCAMA assigns the link to the same time slot with a low contention priority in the same time slot of the current frame. Otherwise, the link is assigned to another random time slot in the current frame with a high priority. In [119], the authors propose a Q-learning algorithm which allow nodes dynamically select link configurations, including channel bandwidth, modulation and coding schemes, guard interval and level of frame aggregation. The node takes these four configurations as states, and the change of these four configurations for next time slot as actions. The reward is given to nodes based on bit error rate.

In [120], the authors consider a dynamic multi-channel access problem. Nodes have to choose a channel without knowing the channel condition in every time slot. Their aim is to maximize the number of successful transmissions. The authors model the problem as a Partially Observable Markov Decision Process (POMDP) and implement a deep Q-network. The authors in [121] address a similar system as [120] where they employ a multi-user deep Q-network to determine the action of nodes in order to share multiple channels efficiently. In [122], the authors demonstrate how multi-user Q-learning algorithm can be used for channel selection over a small network comprising of two users and two channels. In [123], the authors propose a cooperative Q-learning algorithm for cognitive wireless networks. Nodes use the condition of the primary channel, their queue occupancy and buffer capacity as the state. The actions of nodes are whether to remain idle, sense the channel or transmit in each time slot. The authors claim that unlike multiple user Q-learning algorithm where agents always share information, their cooperative Q-learning algorithm only requires agents to share information periodically. In [124], the authors implement a QV-learning algorithm for cognitive wireless networks. They assume the primary user randomly chooses to use its licensed channel in each time slot. A jammer will randomly choose to jam the same channel again, or a new channel. Secondary users
learn to choose a control channel and a data channel while avoiding being jammed by the jammer. The authors in [125] propose a two-stages reinforcement learning algorithm under a multi-armed bandits model for secondary users in cognitive networks. The secondary users firstly learn to predict the channel occupancy for selecting a channel to sense before transmitting. The secondary users also learn to predict the traffic pattern of primary user to predict the possible idle duration of a channel.

The authors in [126] improve slotted Aloha with a stateless Q-learning algorithm. Nodes in a wireless sensor network learn to select slots to avoid collision. Similarly, in [127], the Q-learning algorithm for a wireless sensor network allows nodes to learn to adjust their active time according to their traffic load.

<table>
<thead>
<tr>
<th>Reference</th>
<th>Objective</th>
<th>Framework and Solution</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Karmakar et al.[119]</td>
<td>Link configurations selection.</td>
<td>MDP, Q-learning</td>
<td>Wireless networks</td>
</tr>
<tr>
<td>Cui et al. [106]</td>
<td>Obtain the optimal link schedule</td>
<td>A map that represents node density on a grid, Neural network</td>
<td>Device to device network</td>
</tr>
<tr>
<td>Xu et al. [107]</td>
<td>Improve transmission reliability</td>
<td>MDP, Finite backward induction algorithm</td>
<td>Wireless networks</td>
</tr>
<tr>
<td>Ortiz et al. [108]</td>
<td>Improve energy efficiency</td>
<td>MDP, Multi-agent Q-learning</td>
<td>Energy-harvesting two hop wireless networks</td>
</tr>
<tr>
<td>Yu et al. [109]</td>
<td>Avoid collision and improve transmission reliability</td>
<td>MDP, Deep-reinforcement learning</td>
<td>Wireless networks</td>
</tr>
<tr>
<td>Bayat-Yeganeh et al. [115]</td>
<td>Obtain the optimal schedule</td>
<td>MDP, Q-learning</td>
<td>Wireless networks</td>
</tr>
<tr>
<td>Beheshtifard et al. [116]</td>
<td>Obtain the optimal schedule</td>
<td>Conflict graph, Learning automata</td>
<td>Multi-hop wireless networks</td>
</tr>
<tr>
<td>Stahlbuhk et al. [117]</td>
<td>Obtain the optimal schedule</td>
<td>Combinatorial multi-armed bandits, Greedy algorithm</td>
<td>Wireless networks</td>
</tr>
<tr>
<td>Yi et al. [118]</td>
<td>Obtain the optimal schedule</td>
<td>Conflict graph, Heuristic</td>
<td>Wireless networks</td>
</tr>
</tbody>
</table>
2.4 Discussion

In summary, this chapter has presented legacy MACs and various technologies that help improve the capacity of wireless networks, including multiple channels, adaptive array antennas, MIMO, SIC and IBFD. An IBFD radio has a strong potential to increase network capacity without any extra requirement in bandwidth. Hence, this chapter has presented a number of existing works that aim to exploit IBFD radios.

In addition, this chapter has also presented a number of prior works that aim to design MACs with learning functionalities. The reviewed works, however, have a number of limitations:

<table>
<thead>
<tr>
<th>Authors</th>
<th>Problem Statement</th>
<th>Solution Scheme</th>
<th>Network Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wang et al. [120]</td>
<td>Obtain the optimal schedule</td>
<td>POMDP, Deep Q-neural network</td>
<td>Multichannel wireless networks</td>
</tr>
<tr>
<td>Naparstek et al. [121]</td>
<td>Obtain the optimal schedule</td>
<td>POMDP, Multi-user deep Q-neural network</td>
<td>Multichannel wireless networks</td>
</tr>
<tr>
<td>Li et al. [122]</td>
<td>Channel selection</td>
<td>POMDP, Multi-user Q-learning</td>
<td>Cognitive networks</td>
</tr>
<tr>
<td>Emre et al. [123]</td>
<td>Channel selection and improve energy efficiency</td>
<td>POMDP, Cooperative Q-Learning</td>
<td>Cognitive networks</td>
</tr>
<tr>
<td>Singh et al. [124]</td>
<td>Avoid jamming</td>
<td>POMDP, QV-learning</td>
<td>Cognitive network</td>
</tr>
<tr>
<td>Raj et al. [125]</td>
<td>Improve spectral efficiency</td>
<td>POMDP, Q-learning</td>
<td>Cognitive networks</td>
</tr>
<tr>
<td>Chu et al. [126]</td>
<td>Avoid collisions</td>
<td>MDP, Stateless Q-learning</td>
<td>Wireless sensor networks</td>
</tr>
<tr>
<td>Liu et al. [127]</td>
<td>Improve QoS</td>
<td>MDP, Q-learning</td>
<td>Wireless sensor networks</td>
</tr>
</tbody>
</table>

Table 2.2: A comparison of learning based MACs

In summary, this chapter has presented legacy MACs and various technologies that help improve the capacity of wireless networks, including multiple channels, adaptive array antennas, MIMO, SIC and IBFD. An IBFD radio has a strong potential to increase network capacity without any extra requirement in bandwidth. Hence, this chapter has presented a number of existing works that aim to exploit IBFD radios.

In addition, this chapter has also presented a number of prior works that aim to design MACs with learning functionalities. The reviewed works, however, have a number of limitations:
1. The majority of full-duplex link scheduling works focus on enabling full-duplex transmissions, e.g., [86, 88, 89] and minimizing interference, e.g., [91, 92, 96, 97, 97–100]. However, no work has considered minimizing completion time.

2. The goal of works such as [29, 128–131] is to derive the maximum number of concurrent links. These works, however, assume the channel condition is fixed or has been estimated before transmissions. In practice, the channel gain is likely to vary over time and space. In addition, requiring nodes to estimate the channel before each transmission is expensive. A promising direction is to design MACs with learning capabilities such as [107, 119]. However, these MACs have not considered full-duplex links.

3. The DoF model has been widely used to schedule MIMO links, e.g., [54, 78, 80, 81]. However, none of them have utilized the DoF model to schedule nodes with an IBFD radio. In addition, these works do not consider random traffic loads or time varying queue length.

In the next chapter, this thesis presents three heuristic scheduling algorithms which aim to minimize the completion time of links in centralized wireless networks. These three algorithms are able to construct an optimal schedule in which each link has its own activation duration and data rate.
Minimizing Completion Time

As shown in Chapter 2, existing full-duplex link scheduling works only focus on setting up bi-directional or relay transmissions [86, 88, 89], and aim to minimize interference [91, 92, 96, 97, 97–100]. None of them have considered the Minimum-Time Links Scheduling Problem (MTLSP) problem [128].

MTLSP consists of two sub-problems [129, 130]: 1) select the set of links that can be activated concurrently, and 2) set their activation duration. To solve MTLSP, this thesis extends the usage of the concept ‘affectedness’ from [29], which is a metric for selecting links that can be activated in the same time slot.

Unlike prior works shown in Chapter 2, this chapter considers algorithms that assign concurrently active links with a different activation time and data rate. It also considers interference across multiple cells. Their aim is to minimize the transmission completion time of packets in a dense WLAN where nodes/stations are equipped with an IBFD radio. In this context, controllers play a critical role and are in need of a scheduler that is able to drain the queue of links quickly. Consider the example shown in Figure 3.1. Two APs are connected to a controller and six clients. Both APs and clients are equipped with IBFD radio. The controller is responsible for constructing the optimal schedule in which all transmissions finish in minimum time. As shown previously by Figure 1.3 in Chapter 1, multiple schedules
are available. The prior works \cite{29, 128–130} will give the schedule b) in which each
time slots contains as many as links as possible. This chapter aims to develop a
scheduling algorithm that allows the controller to construct the schedule c). With
the scheduling algorithm, the controller will be aware that although the second time
slot can contain two links, the completion time can be reduced by scheduling link 1
to 3 and link 6 to 5 in two time slots.

Figure 3.1: A dense WLAN consists two cells. Two APs are connected to a central
controller. Full-duplex links are indicated by a double headed blue arrow.

Henceforth, this chapter makes the following contributions:

1. This chapter presents three novel link scheduling algorithms. Given a set of
links with a number of buffered packets, the aim is to drain all packets from
these links in minimum time. Moreover, once a link finishes transmission,
another link is able to start transmission, assuming acceptable interference
from active links. For the first algorithm, aka Algorithm-1, it only enables
full-duplex transmissions whenever possible. However, Algorithm-2 utilizes
full-duplex transmissions only if doing so leads to a reduction in completion
time. Lastly, Algorithm-3 further improves on Algorithm-2 where it greedily
finds the best SINR threshold or data rate for scheduled links.

2. This chapter proposes for the first time algorithms that adopt the concept of
‘affectedness’ \cite{29} for scheduling both half-duplex and full-duplex links. Unlike
past works, e.g., [29] and [129], these algorithms consider three types of interference: 1) self, 2) cross, and 3) exogenous. Compared to [29] and [129], they also consider links with different amounts of data. The proposed algorithms are also able to add a set of links at any time instead of on a slot-by-slot basis, and allow links to have different data rates.

3. The studies in Section 3.5 consider the impact of different node densities and transmission power levels on link schedules; both of which govern the interference experienced by nodes, and hence, their data rates or transmission times. In addition, the impact of different SINR thresholds is also evaluated, which affect the data rate employed by a link given its SINR value. The results in section 3.5 show Algorithm 1 has the second best average performance, with a reduction in completion time of around 40% as compared to having all links transmit individually. Algorithm 2 performs better than Algorithm 1 if the interference between links is strong. Algorithm 3 has the best average performance in all scenarios but incurs the longest computation time.

3.1 Preliminaries

Denote a set of APs as $\mathcal{AP} = \{ap_1, ap_2, ap_3, \ldots, ap_{|\mathcal{AP}|}\}$, and a set of clients, $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_{|\mathcal{C}|}\}$. Both APs and clients are equipped with an IBFD radio. These APs are managed by a controller. Specifically, the controller is responsible for determining the transmission schedule of each AP and client. Moreover, it is aware of the queue corresponding to each link. This queue information is then used by the proposed algorithms, which are run by the controller to determine a transmission schedule.

The set of directed links is denoted as $\mathcal{L} = \{l_1, l_2, l_3, \ldots, l_{|\mathcal{L}|}\}$. Define $l_i(s, r)$, where $s$ and $r$ are respectively the sender and receiver of link $l_i$. Let $P_{wi}$ denote the received power at the receiver of link $l_i$ when the transmitter of link $l_w$ transmits. Hence, for a given link $l_i$, when the transmitter of link $l_i$ transmits, the received power at the receiver is $P_{wi}$. The controller then schedules the transmission of each link to minimize the completion time.
power at the receiver of link \(l_i\) is denoted as \(P_{ii}\).

In order to calculate the received power, say from the transmitter of link \(l_a\) to the receiver of link \(l_b\), i.e., \(P_{ab}\), the following formula is used,

\[
P_{ab} = P_t G_r G_t \left( \frac{c}{4\pi f d} \right)^2
\]

where \(P_t\) is the fixed transmission power by the transmitter of link \(l_a\). The receive and transmit antenna gain is \(G_r\) and \(G_t\), respectively. The Euclidean distance between the sender of link \(l_a\) and receiver of link \(l_b\) is denoted as \(d\). The carrier frequency is \(f\) and the speed of light is \(c\).

Each link \(l_i\) has a start and end time of \(t_s(l_i)\) and \(t_e(l_i)\), respectively. The transmission time of a link \(l_i\) is therefore,

\[
t_c(l_i) = t_e(l_i) - t_s(l_i) = \frac{q_i}{R_i}
\]

where \(q_i\) is an integer number representing the amount of data to be transmitted over link \(l_i\). The symbol \(R_i\) represents the data rate; its exact value is defined later.

Define \(S\) as a set or a schedule containing valid links. Specifically, a link is called valid if it satisfies the following definition:

**Definition 1.** A link \(l\) is valid if both of the following cases are true: \(t_s(l) \geq 0\) and \(t_e(l) \geq t_s(l)\).

Let \(L(t)\) be a set of valid links that are transmitting at time \(t\). Specifically, given a schedule \(S\), \(L(t)\) equals to \(\{l_i \mid t_s(l_i) \leq t \leq t_e(l_i), l_i \in S\}\). In other words, the function \(L(t)\) returns those links in the set \(S\) with a start and end time that overlap with time \(t\).

For a given link \(l_i\), its SINR is defined as,

\[
SINR_i = \frac{P_{ii}}{\sum_{w \in L(t)} P_{wi} + N_o}
\]
where the denominator comprises of the ambient noise $N_o$ and the sum of interference from the transmitter of active links; i.e., it is the sum of received power from the transmitter of link $w \in L(t)$ to the receiver of link $l_i$. Recall that $P_{ti}$ denotes the received power at the receiver of link $l_i$ when the transmitter of link $i$ transmits with power $P_t$. The data rate $R_i$ of link $l_i$ is dependent on its SINR; see Table 3.1. A link is considered collision-free if its SINR is greater than or equal to 4 dB.

Table 3.1: SINR thresholds and their corresponding data rate [1]

<table>
<thead>
<tr>
<th>Thresholds (dB)</th>
<th>Data Rate $R_i$ (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$4 \leq \text{SINR} &lt; 6$</td>
<td>6</td>
</tr>
<tr>
<td>$6 \leq \text{SINR} &lt; 8$</td>
<td>9</td>
</tr>
<tr>
<td>$8 \leq \text{SINR} &lt; 10$</td>
<td>12</td>
</tr>
<tr>
<td>$10 \leq \text{SINR} &lt; 12$</td>
<td>18</td>
</tr>
<tr>
<td>$12 \leq \text{SINR} &lt; 16$</td>
<td>24</td>
</tr>
<tr>
<td>$16 \leq \text{SINR} &lt; 20$</td>
<td>36</td>
</tr>
<tr>
<td>$20 \leq \text{SINR} &lt; 21$</td>
<td>48</td>
</tr>
<tr>
<td>$\text{SINR} \geq 21$</td>
<td>54</td>
</tr>
</tbody>
</table>

A key concept used in this paper is affectedness [29]. As it will be shown later, affectedness is used to determine whether a set of links can transmit concurrently. Formally, the affectedness of link $l_v$ is defined as

$$ A(l_v, L(t)) = \beta \sum_{w \in L(t)} P_{wv} + N_o \over P_{vv} $$

(3.4)

where $\beta$ is the SINR threshold, $N_o$ is the ambient noise and $\sum_{w \in L(t)} P_{wv}$ is the total interference from other simultaneously activated links. Note that unlike [29], the definition of total interference used in this chapter is different due to the use of IBFD radios. Specifically, the total interference suffered by a link $l_v$ includes:

1. Self-interference – this occurs if another link $l_w \in L(t)$ forms a bi-directional full-duplex transmission with $l_v$; in Figure 3.2(a), node A and B have formed a full-duplex link with each other. Consequently, these links interfere with one another. All nodes are assumed to have perfect self-interference cancellation abilities.
2. **Cross-interference** – this occurs when there is another link $l_w \in L(t)$ that forms a “relay” full-duplex transmission with $l_v$; from Figure 3.2(b), the transmission from node-C may interfere with the reception at node-D. The dotted line represents cross-interference. The cross-interference from node-C to node-D is calculated by Equ. (3.1), with node-C as the transmitter and node-D the receiver. Then, the received power from node-C is considered as cross-interference.

3. **Exogenous** – this is the interference from active links emanating from adjacent basic service sets. In Figure 3.2(c), the reception at node-C and node-D is respectively interfered by the transmission from node-A and node-B.

![Interference scenarios](image)

Figure 3.2: Interference scenarios: (a) self, (b) cross, (c) exogenous.

Lastly, the remainder of this chapter makes use of the following definition:

**Definition 2.** A link $l_v$ can co-exist with the links in the link set $L(t)$ if the condition $A(l_v, L(t)) \leq 1$ is true. Otherwise, there is too much interference for link $l_v$ to co-exist with the links in the set $L(t)$.

### 3.2 Problem Definition

The aim is to construct a schedule $S$ containing all links in the set $L$, where each link in the schedule $S$ is valid, and the transmission completion time, i.e., $t_e(l)$, of the last scheduled link $l$ in the schedule $S$ is minimum. Formally, the problem at hand is

$$
\min \left[ \max \{t_e(l_i) \mid l_i \in S \} \right] \quad (3.5)
$$
To illustrate the said problem and notation, consider Figure 3.3. The schedule starts at $t = 0$. Its completion time is $\max\{t_c(l_i) \mid l_i \in \{l_1, l_2, \ldots, l_6\}\} = t_c(l_6)$. Initially, all links have an undefined start and end time. This means they do not belong to the schedule $S$. At time $t = 0$, several links are added into the schedule $S$. All these links must satisfy Definition 2. In this example, we see that links $l_1$, $l_2$, $l_3$ and $l_4$ can co-exist with each other. Hence, they have a start time of $t_s(l_i) = 0$, where $i = \{1, 2, 3, 4\}$. Assume link $l_3$ finishes its transmission first. At this point, there is an opportunity to add another link. However, doing so may cause the SINR or data rate of existing links to degrade. For this reason, after adding a new link, all links in the set $S$ must continue to satisfy Definition 2. In this example, we see that link $l_6$ can be added successfully after link $l_3$ completes. Similarly, we see link $l_5$ is added after link $l_1$ has transmitted all its data. The start time of link $l_5$ is set to $t_s(l_5) = t_c(l_1)$. As for link $l_6$, its start time is $t_s(l_6) = t_c(l_3)$.

![Figure 3.3: An example link schedule.](image)

From the above example, it can be observed that the set of links and the resulting interference affect the completion time. This is because different sets of links will yield different interference, which impact the SINR or data rate of simultaneously active links. Moreover, a set of active links may delay new links from being added due to excessive interference.

Note that a special case of the problem at hand is to assume all transmissions complete at slot boundaries. In this case, the problem has been proven to be NP-hard; see [129]. Specifically, assume all links have the same data length and have the same data rate or SINR threshold. Then the problem is to derive a schedule with minimum length such that all links are activated once. This is exactly the NP-hard
problem in [129]. However, the problem in this chapter is more general where links have different data lengths and a link can be added into a schedule whenever another link completes its transmission. By contrast, in [129], links can only be added at slot boundaries. That is, all links within a slot must finish transmission before another set of links start. As it will be shown in Section 3.4, relaxing this restriction leads to smaller completion times.

Lastly, the work outlined in this chapter remains applicable when traffic arrives randomly, and links have different amounts of data. Let $Q_1(t)$ and $Q_2(t)$ be the queue of two APs at time $t$. The goal is thus to derive a schedule that transmits packets in $Q_1(t)$ and $Q_2(t)$ in the shortest possible time. This is important because a fast completion time means a high throughput or network capacity. Once the packets in $Q_1(t)$ and $Q_2(t)$ have a transmission schedule or time, then we can consider the next batch of unscheduled packets at time $t + 1$. Note that $Q_1(t+1)$ and $Q_2(t+1)$ contain a random number of unscheduled packets that have arrived in the period $[t, t+1]$ according to some traffic load distribution. Thus we have the same problem at time $t + 1$, which is to calculate a schedule for newly arrived packets. For this reason, we only need to consider scheduling a set of links with some random amounts of data.

3.3 Scheduling Algorithms

This section presents three novel algorithms that are run at the controller of a WLAN. Their basic idea is to add one or more links into the schedule whenever a link finishes subject to links meeting their SINR requirement. The second algorithm further considers whether adding a link reduces the overall completion time. The last algorithm also identifies the best data rate for each link when it is activated along with other active links.
3.3.1 Algorithm 1

Algorithm 1 aims to maximize simultaneous transmissions. It takes the link set $L$ as input. Firstly, it selects the links that have a higher SINR than their chosen threshold and includes them into the set $L_g$. The rationale here is that these links have the best chance of supporting full-duplex transmissions. Then, Algorithm 1 adds as many links as possible from the set $L_g$ into the final schedule $S$ whenever a valid link ends its transmission. Links that cannot be activated concurrently are then scheduled to transmit one after another.

Algorithm 1 operates as follows. In line-1, it initializes three empty sets: (i) $S$, which records the final schedule, (ii) $L_g$, which stores possible full-duplex links, and (iii) $L_h$, which contains links capable of half-duplex transmissions only. In lines 2-6, the function $\text{SNR}(l_i)$ calculates the SNR for each link in $L$; i.e., each link transmits by itself without interference. If the obtained SNR value is less or equal to a given threshold $\beta$, then $l_i$ cannot transmit concurrently with other links and it is added into the set $L_h$. Otherwise, the link $l_i$ will be added into the set $L_g$.

The goal of lines 8-18 is to add as many concurrent links as possible when a link finishes transmission. In line-9, the set $\Delta$ contains the end time of all links in $S$ sorted in increasing order. The $n$-th element of $\Delta$ is denoted as $\Delta(n)$. The set $S^*$ includes all active links in the time period $\Delta(n)$ to $\Delta(n+1)$. Lines 11 to 17 iterate through links in $L_g$. The function $\text{Coexist}(S^*, l_i, \beta)$ determines whether the links in $S^*$ satisfy Definition 2 after adding the link $l_i \in L_g$ into $S^*$. If all links satisfy Definition 2, then $l_i$ can be added into the schedule $S$ and function $\text{Coexist}(S^*, l_i, \beta)$ returns true. Otherwise, the function returns false. If $\text{Coexist}(S^*, l_i, \beta)$ is true, the function $\text{AssignParams1}(l_i)$ gives the link $l_i$ a start time of $t_s(l_i) = \Delta(n)$. Then, $\text{AssignParams1}(l_i)$ determines the SINR value for link $l_i$ according to Equ. (3.3) and assigns link $l_i$ a data rate $R_i$ according to Table 3.1. $\text{AssignParams1}(l_i)$ also sets $t_e(l_i) = \frac{q_i}{R_i} + t_s(l_i)$ as the end time for link $l_i$; i.e., the end time of link $l_i$ is its start time plus the time required to transmit $q_i$ bits. This link is then added into
the schedule $S$ (line-14). Finally, link $l_i$ is removed from $L_g$ and the while loop (lines 8 to 18) continues until $L_g$ becomes empty. Lastly, before returning the set of links in $L_b$, the function $\text{AssignParams2}(L_b)$ assigns a start and end time, and a data rate to each link in the set $L_b$. These links transmit one after another at the highest possible data rate.

\begin{algorithm}
\begin{algorithmic}[1]
\State \textbf{Data:} Unscheduled links set $L$
\State \textbf{Result:} Scheduled links set $S$
\State $S = L_g = L_b = \emptyset$;
\For {each link $l_i \in L$}
\If {$\text{SNR}(l_i) \leq \beta$}
\State $L_b \cup l_i$;
\Else
\State $L_g \cup l_i$;
\EndIf
\EndFor
\State $n = 0$;
\While {$L_g \neq \emptyset$}
\State $\Delta = \{t_e(l_a), t_e(l_b), \ldots, t_e(l_m) \mid t_e(l_a) < t_e(l_b) < \ldots < t_e(l_m) ; l_a, l_b, \ldots, l_m \in S\}$;
\State $S^* = \{l_i \mid \Delta(n) \leq t_e(l_i) \leq \Delta(n+1) \lor \Delta(n) \leq t_e(l_i) \leq \Delta(n+1), l_i \in S\}$;
\For {each link $l_i \in L_g$}
\If {$\text{Coexist}(S^*, l_i, \beta) = \text{true}$}
\State $\text{AssignParams1}(l_i)$;
\Else
\State continue;
\EndIf
\EndFor
\State $n = n + 1$;
\EndWhile
\State return $S \cup \text{AssignParams2}(L_b)$;
\end{algorithmic}
\end{algorithm}

As an example, consider how Algorithm 1 generates a schedule $S$ for the links shown in Figure 3.4. Clients $C_1$, $C_2$ and $C_3$ are connected to AP $A_1$, and clients $C_4$, $C_5$ and $C_6$ are connected to AP $A_2$. The dotted lines represent possible interference between clients. A thin dotted line means weak interference and a thick dotted line means strong interference. In this example, there are six links; namely $l_1(C_1, A_1)$,
l_2(C_2, A_1), l_3(A_1, C_3), l_4(A_2, C_4), l_5(C_6, A_2), l_6(A_2, C_5). To simplify exposition, assume that if a link suffers from weak interference, its current data rate does not change. On the other hand, if there is strong interference, its data rate drops by $\frac{2}{3}$.

Algorithm 1 will return the schedule shown in Figure 3.5. After the first iteration of the while loop (lines 8 to 18), the function $\text{Coexist}(S^*, l_i, \beta)$ determines that links $l_1$, $l_3$, $l_4$ and $l_5$ can transmit concurrently. Two full-duplex transmissions are formed by links $l_1$ and $l_3$, and also links $l_4$ and $l_5$. The function $\text{AssignParams1}(l_i)$ assigns a start and end time to these four links, as well as a suitable data rate based on their SINR. In this example, assume the data rate remains the same as if the links transmit independently because all clients and APs are assumed to have perfect self-interference cancellation.

Then, the function $\text{Coexist}(S^*, l_i, \beta)$ determines whether links $l_2$ and $l_6$ can transmit concurrently. When client $C_2$ transmits, its signal at $A_1$ will not be interfered by the transmission from $A_2$. Therefore, link $l_2$ is able to retain its current data rate. Unfortunately, link $l_2$ will cause a strong interference towards link $l_6$ because the receiver of link $l_6$ is client $C_5$ which is close to the transmitter of link $l_2$. Assume the data rate drops to $1/3$ of the current data rate. Therefore, the function $\text{AssignParams1}(l_i)$ assigns link $l_2$ with the data rate that this link uses when it transmits by itself; i.e., no interference. Similarly the function $\text{AssignParams1}(l_i)$
assigns link \( l_6 \) with \( \frac{1}{3} \) of the data rate that it would have used if it transmitted independently. However, the overall completion time \( T^{S_1}_e \) is still smaller than that when there are no concurrent transmissions, which ends at time \( T_e \).

![Diagram showing result of Algorithm 1](image)

Algorithm 1 uses the lowest possible SINR threshold, meaning links transmit at the lowest data rate. In some case, their overall completion time may exceed the case where they transmit independently without interference. The next algorithm overcomes this weakness.

### 3.3.2 Algorithm 2

Algorithm 2 has two major differences from Algorithm 1. Firstly, it selects the SINR threshold \( \beta \) according to a base SNR value that corresponds to the case where all links transmit individually, meaning there is no interference. Secondly, it only allows multiple links to transmit concurrently if doing so reduces the overall completion time.

Line 2 of Algorithm 2 sets the concurrent transmission SINR threshold \( \beta \) to the average SNR value of all links in \( L \). This corresponds to all links transmitting individually, where there is no interference. Then, lines 3-7 construct two sets: \( L_b \) and \( L_g \). The set \( L_g \) comprises of all links that satisfy Definition 2. Otherwise, these links are included in the set \( L_b \). The while loop from lines 9 to 29 aim to find sets of type \( a_i \) and the corresponding completion time reduction, denoted as \( t^*_i \). Specifically, each \( a_i \) contains links that can be added into the schedule \( S \) at
time \( t_n \) without violating Definition 2. The variable \( t^*_i \) represents how much the overall completion time can be reduced if all links in \( a_i \) transmit concurrently. The calculation is achieved by the function \( \text{RecordTime}(a_i) \). A larger \( t^*_i \) value means a higher reduction in completion time if all links \( l_i \in a_i \) are added into the schedule \( S \) at \( t_n \) and transmit concurrently. Therefore, the function \( \text{BestCandidate}(A) \) returns the \( a_i \in A \) that yields the highest reduction in overall completion time. Let this set be the set \( a_b \). The function \( \text{AssignParams1}(a_b) \) gives each link \( l_i \) in \( a_b \) a start time \( t_s(l_i) = t_n \). Then, \( \text{AssignParams1}(a_b) \) determines the SINR value for each link \( l_i \) in \( a_b \) according to Equ. (3.3). Next, the function \( \text{AssignParams1}(a_b) \) gives each link \( l_i \) in \( a_b \) a suitable data rate \( R_i \) according to Table 3.1, as well as assigns \( t_e(l_i) = \frac{q_i}{R_i} + t_s(l_i) \) as the end time for each link. Finally, all links in \( a_b \) are included into the schedule \( S \).

As an example, consider how Algorithm 2 generates a schedule for the same example used to illustrate Algorithm 1. Figure 3.6 shows that Algorithm 2 returns a different schedule. During the first iteration of its while loop, Algorithm 2 will find \( a_i = \{l_1, l_3, l_5, l_4\} \) and the overall completion time can be reduced if all these links transmit concurrently. Thus, the result is the same as Algorithm 1. Then, in the second iteration, Algorithm 2 finds \( a_i = \{l_2, l_6\} \). Links \( l_2 \) and \( l_6 \) can transmit concurrently but the overall completion time will not be reduced as compared to the case when link \( l_2 \) and \( l_6 \) transmit individually. Thus, Algorithm 2 rejects \( a_i = \{l_2, l_6\} \) and simply lets link \( l_2 \) transmits by itself. In the next iteration, there is only link \( l_6 \). Thus, Algorithm 2 schedules link \( l_6 \) to transmit right after link \( l_2 \) finishes its transmission. Finally, the overall completion time follows \( T_{e}^{S_2} < T_{e}^{S_1} < T_{c} \).

The foregone example shows that Algorithm 2 is able to find a better schedule as compared to Algorithm 1; it, however, incurs additional computation time; see Section 3.3.4. Also, Algorithm 2 is able to dynamically choose an SINR threshold based on the average SNR of links in \( L \). However, this can reduce the number of concurrent transmissions when Algorithm 2 selects an high SINR threshold, e.g., 12 dB. This situation occurs when links in \( L \) have a high SNR. To overcome this
Algorithm 2: Dynamic SINR threshold $\beta$

**Data:** Unscheduled links set $L$

**Result:** Scheduled links set $S$

1. $S = L_g = L_b = \emptyset$;
2. $\beta = \frac{\sum_{l_i \in L} \text{SNR}(l_i)}{|L|}$;
3. for each link $l_i \in L$ do
   4. if $\text{SNR}(l_i) \leq \beta$ then
      5. $L_b \cup l_i$;
   6. else
      7. $L_g \cup l_i$;
   8. end
4. $n = 0$;
5. while $L_g \neq \emptyset$ do
6.   7. $\Delta = \{t_e(l_a), t_e(l_b), \ldots, t_e(l_m) \mid t_e(l_a) < t_e(l_b) < \cdots < t_e(l_m), l_a, l_b, \ldots, l_m \in S\};$
7.   8. $S^* = \{l_i \mid \Delta(n) \leq t_s(l_i) \leq \Delta(n+1) \lor \Delta(n) \leq t_e(l_i) \leq \Delta(n+1), l_i \in S\};$
8.   9. $L_{g}^* = L_g$;
9. 10. $A = \emptyset$;
10. while $L_{g}^* \neq \emptyset$ do
11.   12. $a_i = \emptyset$;
13.   13. $t^*_i = 0$;
14.   14. for each link $l_i \in L_{g}^*$ do
15.     15. if $\text{Coexist}(S^*, l_i, \beta) = \text{true}$ then
16.       16. $a_i \cup l_i$;
17.       17. $L_{g}^* \setminus l_i$;
18.     18. else
19.       19. continue;
20.     20. end
21.     21. $t^*_i = \text{RecordTime}(a_i)$;
22.     22. $A \cup (a_i, t^*_i)$;
23.   19. end
24.   20. $a_b = \text{BestCandidate}(A)$;
25.   21. $\text{AssignParams1}(a_b)$;
26.   22. $S \cup a_b$;
27.   23. $L_g \setminus a_b$;
28.   24. $n = n + 1$;
11. end
30. return $S \cup \text{AssignParams2}(L_b)$;
3.3. Scheduling Algorithms

Figure 3.6: Result of Algorithm 2. This figure also shows a schedule without concurrent transmissions that ends at time $T_e$.

weakness, the next algorithm will iterate through all SINR thresholds recorded in Table 3.1 to determine whether there exists a data rate that reduces the overall completion time.

3.3.3 Algorithm 3

Algorithm 3 is a slightly modified version of Algorithm 2. There are only two differences as compared to Algorithm 2. Firstly, in lines 2-6, the links $l_i \in L$ are divided into $L_b$ and $L_g$ by the minimum SNR requirement 4 dB. Secondly, Algorithm 3 greedily searches all transmission SINR thresholds value $\beta_i \in \Omega$, where the set $\Omega$ contains the SINR thresholds shown in Table 3.1. This is achieved by an additional loop outside the while loop from lines 13 to 24 in Algorithm 2. Thus, lines 14 to 24 of Algorithm-3 are carried out to find all tuples $(a_i, t^*_i)$ under different $\beta_i$ values.

Note that Algorithm 3 will give the same schedule for the example used to illustrate Algorithm 1 and 2. However, it incurs a higher computation time because it needs to search through all possible SINR thresholds. For more complex scenarios, see Section 3.5, Algorithm 3 is able to find a better result as compared to both Algorithm 1 and 2.

3.3.4 Analysis

The propositions to follow concern the properties of the proposed algorithms.

Proposition 1. The run time complexity of Algorithm 1 is $O(|AP||L|^2)$. 

54
Algorithm 3: Greedy Search

Data: Unscheduled links set \( L \)

Result: Scheduled links set \( S \)

1. \( S = L_g = L_b = \emptyset; \)
2. for each link \( l_i \in L \) do
   3. if \( \text{SNR}(l_i) \leq 4 \text{ dB} \) then
      4. \( L_b \cup l_i; \)
   5. else
      6. \( L_g \cup l_i; \)
   7. end
3. \( n = 0; \)
4. while \( L_g \neq \emptyset \) do
5. \( \Delta = \{ t_e(l_a), t_e(l_b), \ldots, t_e(l_m) \mid t_e(l_a) < t_e(l_b) < \cdots < t_e(l_m), l_a, l_b, \ldots, l_m \in S \}; \)
6. \( S^* = \{ l_i \mid \Delta(n) \leq t_e(l_i) \leq \Delta(n+1) \lor \Delta(n) \leq t_e(l_i) \leq \Delta(n+1), l_i \in S \}; \)
7. \( L^*_g = L_g; \)
8. \( A = \emptyset; \)
9. for each \( \beta_i \) in \( \Omega \) do
10. while \( L^*_g \neq \emptyset \) do
11. \( a_i = \emptyset; \)
12. \( t^*_i = 0; \)
13. for each link \( l_i \in L^*_g \) do
14. if Coexist\( (S^*, l_i, \beta) = \)true then
15. \( a_i \cup l_i; \)
16. \( L^*_g \setminus l_i; \)
17. else
18. continue;
19. end
20. \( t^*_i = \text{RecordTime}(a_i); \)
21. \( A \cup (a_i, t^*_i); \)
22. end
23. \( a_b = \text{BestCandidate}(A); \)
24. \( \text{AssignParams1}(a_b); \)
25. \( S \cup a_b; \)
26. \( L_g \setminus a_b; \)
27. \( n = n + 1; \)
28. end
29. return \( S \cup \text{AssignParams2}(L_b); \)
Proof. Lines 2 to 6 have complexity $O(|L|)$ because they check each link in $L$. Note that function $\text{SNR()}$ has $O(1)$ time complexity. The worst case for Algorithm 1 is when all links in $L$ are included into the set $L_g$, i.e., $|L_b| = 0$, and when each iteration of the while loop (lines 8 to 18) only adds one link into the schedule. In iteration $m + 1$, schedule $S$ contains $m$ links, for $m = 1, 2, \ldots, |L| - 1$. Thus, line 9 and 10 will both have a complexity of $O(m)$ because they search through every link in $S$, or in total, over all iterations of the while loop, the two lines require $(1 + 2 + \cdots + |L| - 1)$ searches, or $O(|L|^2)$. In iteration $m + 1$ of the while loop (lines 8 to 18), the for loop (lines 11 to 18) has $|L| - m$ iterations.

Function $\text{CoExist()}$ has complexity $O(|AP|)$ because $S^*$ must contain $2 \times |AP| - 1$ links in the worst case in order to obtain schedule $S$. Function $\text{AssignParams1()}$ as well as lines 14 and 15 each has a complexity of $O(1)$. Thus, lines 11 to 17 in total, over all iterations of the while loop, are iterated $O(|L| - 1 + |L| - 2 + \cdots + 1)$ times. Since each iteration of the for loop takes $O(|AP|)$, the complexity of lines 11 to 17 is $O(|AP||L|^2)$. Finally, the overall complexity for Algorithm 1 is $O(|L|^2 + |L| + |AP||L|^2) = O(|AP||L|^2)$. Note that function $\text{AssignParams2()}$ has complexity $O(|L_b|)$, and for this case, i.e., $|L_b| = 0$, the function takes $O(1)$.

Proposition 2. The run time complexity of Algorithm-2 is $O(|AP||L|^3)$

Proof. Line 2 as well as lines 3 to 7 have a time complexity of $O(|L|)$. The worst case time for Algorithm 2 is when all links in $L$ belong to set $L_g$, i.e., $|L_b| = 0$ and each iteration of the while loop (lines 9 to 29) inserts only one link into the schedule $S$. The while loop (lines 9 to 29) has $|L|$ iterations, and after iteration $m$, for $m = 1, 2, \cdots, |L| - 1$, the schedule $S$ contains $m$ links. Thus, lines 10 and 11 each takes $O(m)$ to search $S$, or in total, each line takes $O(|L|^2)$. Further, the while loop (lines 14 to 24) iterates for $|L| - m$ times, and in each iteration, the for loop (lines 17 to 22) also has $|L| - m$ iterations. Similar to Algorithm 1, function $\text{CoExist()}$ has a complexity of $O(|AP|)$, and lines 19 and 20 both have complexity $O(1)$. Thus, the for loop (lines 17 to 22) takes $O(|AP||L|^2)$ time for each iteration of the while loop.
loop (lines 14 to 24) or in total, over all iterations of the while loop in lines 9 to 29, has \(O(|AP||L|^3)\) time complexity. Function \textbf{RecordTime}(a_i) has complexity \(O(1)\) because \(a_i\) contains only one link in the worst case for obtaining a schedule, and line 24 has complexity \(O(1)\). Function \textbf{BestCandidate}(A) has complexity \(O(|L| - m)\) because set \(A\) contains \(((|L| - m)|a_i|)\) links and each \(a_i\) contains only one link. As in Algorithm 1, function \textbf{AssignParams1()} and \textbf{AssignParams2()} for this case each has time complexity of \(O(1)\). The remaining lines have complexity \(O(1)\). Finally, Algorithm 2 has complexity \(O(|L|^2 + |AP||L|^3 + |L|^2) = O(|AP||L|^3)\) \(\Box\)

**Proposition 3.** The run time complexity of Algorithms 3 is \(O((|AP| + |\Omega|)|L|^3))\).

**Proof.** Algorithm 3 is a modified version of Algorithm 2. The only difference is that Algorithm 3 searches through all \(|\Omega|\) SINR thresholds instead of using only a single SINR threshold. This means the for loop from lines 13 to 24 forces the secondary while loop from lines 14 to 24 to execute at most \(|\Omega|\) times. The remaining lines are exactly the same as Algorithm 2. Thus, the run time complexity of Algorithm 3 is \(O(|L|^2 + (|AP| + |\Omega|)|L|^3)) = O((|AP| + |\Omega|)|L|^3))\). \(\Box\)

**Proposition 4.** All algorithms produce a schedule \(S\) that ensures all links satisfy their SINR threshold.

**Proof.** In each algorithm, the function \textbf{Coexist}(S*, l, \beta) determines whether a link \(l_i\) can be added into the schedule \(S\) when an active link \(l_w\) completes its transmission. The subset \(S^*\) contains all links that are activated at \(t_c(l_w)\), which is the end time of \(l_w\). When \textbf{Coexist}(S*, l, \beta) returns true, all links in \(S^*\) and link \(l_i\) have an affectedness \(A(l_i, S^*)\) that is less than one. According to Definition 2, all links in \(S^*\) and link \(l_i\) are able to transmit concurrently. In addition, according to Equ. (3.6), all links must meet their SINR threshold, i.e., \(SINR_i > \beta\), when function \textbf{Coexist}(S*, l, \beta) returns true because,

\[
\frac{P_{wv}}{\sum_{w \in L(t)} P_{wv} + N_o} = SINR_i = \frac{\beta}{A(l_i, S^*)}
\]
The value of $\beta$ is always larger or equal to 4 dB, which is the minimum requirement for links to coexist. The affectedness $A(l_i, S^*)$ must be less than one because the function $\text{Coexist}(S^*, l_i, \beta)$ returned true. Therefore, the condition $\text{SINR}_i = \frac{\beta}{A(l_i, S^*)} > \beta$ is true. Thus, all links in the schedule are able to successfully transmit because they must have an SINR value that is at least 4 dB.

The last proposition outlines a key relationship between the time gained from scheduling concurrent transmissions and the time loss due to the increased in transmission time resulting from higher interference.

Let $\tau_i$ denote the transmission time when link $i$ transmits by itself; i.e., this is the transmission time corresponding to the data rate used when there is no interference. Let Scheduler-0 returns a schedule where links transmit one after another by themselves. Denote $T^0_m$ as the completion time of $m$ links as computed by Scheduler-0. Next, consider an arbitrary scheduler referred to as Scheduler-z that schedules the same $m$ links. Let the completion time of these $m$ links be $T^z_m$. Consider the scenario where Scheduler-z activates link $i$ concurrently with $m-1$ links. The term $\tau_i$ denotes the saved time. That is, the completion time $T^z_m$ is now potentially $\tau_i$ shorter with respect to $T^0_m$. As an example, consider two links with transmission time $\tau_1$ and $\tau_2$. For simplicity, assume $\tau_1 = \tau_2$. Using Scheduler-0, we have $T^0_2 = \tau_1 + \tau_2$. However, if both links are scheduled concurrently, then the completion time is $\tau_1$. In other words, the completion is saved by $\tau_2$. Equivalently, $T^0_2$ is reduced by $\tau_2$ time.

Let $S^+_m$ be the total saved time when $m$ links have been added into the schedule. For example, if there are three links with transmission time $\tau_1 > \tau_2 > \tau_3$, and we schedule link-2 and link-3 to transmit concurrently with link-1, then $S^+_3$ equals to $\tau_2 + \tau_3$.

Let $\phi_i \geq 1$ be a multiplicative factor that indicates the increased in transmission time when a link is scheduled with another link. As an example, consider two links that are scheduled together and also interfere with each other. Then we may have $\phi_1 = 1.1$, where $\phi_i \tau_i$ means the transmission time has increased by 10%. Equiva-
lently, \((\phi_i - 1)\tau_i\) is the extra transmission time incurred due to a lower data rate being used to combat the increased interference. Define \(S_m^-\) as the sum increased in transmission time after \(m\) links have been added into the schedule. Let \(S\) be a schedule with \(m\) links. Formally, we have,

\[
S_m^- = \sum_{i \in S} (\phi_i - 1)\tau_i
\]  

(3.7)

We then have the following proposition.

**Proposition 5.** Assume Scheduler-0 and Scheduler-z select \(m\) links in the following order: \(l_1, l_2, \ldots, l_m\). If Scheduler-z ensures \(S_m^+ \geq S_m^-\), then we have \(T_m^z \leq T_m^0\).

**Proof.** Initially, the schedule \(S\) of both schedulers only contains \(l_1\). Hence, the inequality \(T_1^z \leq T_1^0\) is true, where \(S_1^+ = S_1^- = 0\). Assume \(S_{m-1}^+ \geq S_{m-1}^-\) when Scheduler-z picks \(l_m\). There are three cases to consider.

**Case-1:** link \(l_m\) can co-exist with the links in \(S\) and the data rate of all links remains the same. Hence, we have \(T_m^z < T_m^0\) because some links are scheduled concurrently with other links. Note, in this case, \(S_m^+ = S_{m-1}^+ + \tau_m\) and \(S_m^- = 0\). **Case-2:** links in \(S\) and \(l_m\) cannot co-exist with one another due to strong interference. Hence, link \(l_m\) must be scheduled to transmit independently. In this case, the inequality \(T_m^z \leq T_m^0\) holds as there is no gain in saved time and scheduled links have the same data rate, meaning \(S_m^+ = S_{m-1}^+\) and \(S_m^- = S_{m-1}^-\). **Case-3:** in this case, all links in \(S\) suffer increased weak interference, meaning \(\phi_i \geq 1\) for all \(i \in S \cup l_m\). If \(S_m^+ - S_m^- < 0\), then adding link \(l_m\) results in a total increase in transmission time that exceeds the saved time. Equivalently, the resulting schedule will exceed the one computed by Scheduler-0. So we must have \(S_m^+ - S_m^- \geq 0\). This implies \(T_m^z \leq T_m^0\), as desired. \(\square\)

### 3.4 Evaluation

The evaluation methodology is to determine factors that influence the transmission completion time. In particular, the evaluation is not concerned with protocol be-
haviours or channel errors. To this end, all three algorithms are implemented in
C#. All APs and clients are randomly placed on a square area of size 2500 m².
Each AP and client pair consists of an up and down link. Each link is initialized
with random amounts of data drawn from a Gaussian distribution with a mean of
15 and variance of five at each iteration. The unit of data length is MBytes. The
input link set \( L \) contains all links sorted in an ascending order based on their data
length. This chapter studies the impact of the following parameters:

1. **Node density.** This is the ratio between the number of APs and the number of
   clients, denoted as \( \frac{|C|}{|AP|} \), which ranges from one to 15 with an interval of one.

2. **Transmission power.** The transmission power ranges from 1 to 25 mW with
   an interval of 1 mW.

3. **SINR threshold \( \beta \).** The value of \( \beta \) is chosen from Table-3.1. This parameter
   is only of concern when evaluating the performance of Algorithm 1 because
   Algorithm 2 and 3 choose an SINR threshold \( \beta \) automatically.

A reference algorithm is also created to benchmark against the proposed algo-
rithms. This reference algorithm, labelled as Algorithm-SDT, models the algorithms
in [29][129][128] and [130] where links are scheduled on a slot-by-slot basis. More-
over, no new links are added when a link completes its transmission. Links have the
same start time. Also, these links have the same data rate. In the first two experi-
ments, Algorithm-SDT uses an SINR threshold of \( \beta = 4 \) dB, meaning it is the lowest
possible data rate. In the experiment reported in Section 3.5.3, Algorithm-SDT as-
signs the highest possible data rate from Table 3.1 that allows them to transmit
simultaneously given the interference from other active links.

In the sequel, for each network topology \( k \), for the schedule where links transmit
one after another, its completion time is denoted as \( T_{ck} \). On the other hand, for a
given schedule \( S_i \) computed by Algorithm \( i \), where \( i = \{1, 2, 3, SDT\} \), its completion
time is denoted by \( T_{c}^{S_i,k} \). In all experiments, the following metrics are recorded:
3.5 Results

1. Average completion time reduction ($\Delta$). That is,

$$\Delta = \frac{1}{N} \sum_{k=1}^{N} \left( 1 - \frac{T_{S_i}}{T_{c_k}} \right)$$  \hspace{1cm} (3.8)

The integer $N$ is the number of tested network topologies which is set to ten thousands.

2. Maximum completion time reduction ($\Delta^+$). This is the maximum reduction time over all tested network topologies. It is defined as

$$\Delta^+ = \max \left( 1 - \frac{T_{S_i}}{T_c} \right)$$  \hspace{1cm} (3.9)

3. Minimum completion time reduction ($\Delta^-$). This is the minimum reduction in completion time over all tested topologies. Specifically,

$$\Delta^- = \min \left( 1 - \frac{T_{S_i}}{T_c} \right)$$  \hspace{1cm} (3.10)

4. Average computation time. Each experiment is run on a computer with an Intel i7-6700 and 16 GB RAM, and their running time is recorded.

3.5 Results

The subsequent sections present results from experiments in scenarios with different node densities, transmission power levels, and SINR thresholds.

3.5.1 Node Density

The number of APs is set to five. The transmission range is set to 15 meters. The SINR threshold for Algorithm 1 and Algorithm-SDT is 4 dB. In Figure 3.7 and 3.8, it can be observed that both $\Delta$ and $\Delta^+$ decrease when the node density increases. The reason is that the proposed algorithms schedule multiple links to
transmit concurrently. However, the maximum number of concurrent transmissions is bounded by the number of APs because each AP can only support one up and one down link at a time. In addition, the total number of links increases with node density. Therefore, the quantities $\Delta$ and $\Delta^+$ can only decrease with node density given the higher interference experienced by links.

![Figure 3.7: Average completion time reduction versus node density.](image)

Algorithm 3 achieved the best $\Delta$, $\Delta^+$ and $\Delta^-$ value; see Figures 3.7, 3.8 and 3.9. The $\Delta$ value of Algorithm 3 is about 33% initially and reduces to about 28% when the node density is larger than five. The maximum reduction in completion time, i.e., $\Delta^+$, of Algorithm 3 is about 51% initially and reduces to about 38% with increasing node density. The $\Delta^-$ value of Algorithm 3 fluctuates between 0.0006% and 16%. Algorithm 3 has better performance than Algorithm 1 because it greedily searches through all SINR values. Algorithm 3 also only allows concurrent transmissions if the completion time of links is no longer than when they transmit one by one. Thus, we observe that the $\Delta^-$ value of Algorithm 3 does not contain any value below zero.

Note that the fluctuations seen in Figure 3.8 and 3.9 are due to the use of arbitrary network topologies, which give rise to non-trivial interference relationships between links. All experiments are conducted over 10000 arbitrary topologies, links may
3.5. Results

Figure 3.8: Maximum completion time reduction versus node density.

Figure 3.9: Minimum completion time reduction versus node density.
be placed far apart, meaning they do not interfere with one another significantly. Consequently, their high SINR allows them to use a high data rate. As a result, they complete their transmission quickly. On the other hand, links could be placed very closely together. Hence, they may interfere with each other significantly which means their data rate is likely to be low and they require a longer completion time.

Algorithm 1 achieves the second best average reduction in completion time or $\Delta$ value; it is about 29% initially and reduces to about 25% after the node density reaches five. Algorithm 1 has a worse $\Delta^+$ than Algorithm 3. The maximum reduction, i.e., $\Delta^+$, of Algorithm 1 is 49% initially and about 2.5% lower than Algorithm 3 when the node density is eleven. The reason is that Algorithm-1 uses the lowest SINR value level of 4 dB as a threshold and allows links to transmit concurrently whenever it is possible to do so. Consequently, Algorithm 1 schedules more concurrent links, which has a positive impact on both $\Delta$ and $\Delta^+$. However, as Algorithm 1 allows links to transmit concurrently, doing so may cause a reduction in the data rate of some links. Therefore, the $\Delta^-$ value of Algorithm 1 is $-8\%$ when the node density is eight. Note that when using Algorithm-1, links scheduled to transmit together may experience significant interference. If their data rate is low, then the completion may be longer than the schedule where links transmit one by one and at the highest possible data rate. The maximum $\Delta^-$ value is 3%, which is thirteen percentage points lower than Algorithm 3.

Algorithm-SDT achieved the third best $\Delta$ when the node density is less than eight; its $\Delta$ value is 29% initially and reduces to 15% when the node density is fifteen. Algorithm-SDT achieved nearly identical $\Delta^+$ as Algorithm 1, which recorded a reduction of 48% initially and reduces to about 36% when the node density is fifteen. The reason why Algorithm-SDT has worse performance in terms of $\Delta$ as compared to Algorithm 1 is because Algorithm-SDT assigns the same data rate and activation time to links belonging to the same subset. A link may have a high SINR but it is assigned a low data rate because other links in the subset have a low SINR. A link may also have to remain active longer than needed because other links in the
same subset have not finished transmission. Both scenarios have a negative effect on reducing the overall transmission completion time. Thus, Algorithm-SDT performs worse than Algorithm 1 in terms of $\Delta$. The same reason also causes Algorithm-SDT to have multiple $\Delta^-$ with negative values; i.e., their completion time is worse than the case where links transmit on their own. However, links can also have a similar SINR value and data rate in each subset because their parameters are generated randomly. In this situation, Algorithm-SDT can have a similar or even the same performance as Algorithm 1.

Algorithm 2 has the worst $\Delta$ when the node density is less than seven, which is 24% initially and reduces to 17% when the node density is seven. The reason is that Algorithm 2 uses the average SNR value of links in $L$ as the SINR threshold $\beta$. The SINR threshold $\beta$ chosen by Algorithm 2 will be higher than Algorithm 1 and Algorithm-SDT because they use the lowest SINR value in Table 3.1. Thus, Algorithm 2 allows fewer links to transmit concurrently as compared to other algorithms; this fact causes Algorithm 2 to have a longer completion time. However, the performance of Algorithm 2 is better than Algorithm-SDT in terms of $\Delta$ when the node density is larger than seven. The reason is that Algorithm-SDT may assign a link with a lower data rate than the one it can support because of other links with a low SINR in the same subset. A link may also need to have the same activation time as these links. These factors cause the overall completion time to increase and their impact becomes more pronounced with higher node densities due to the increased interference. Therefore, the performance of Algorithm-SDT is lower than Algorithm 2 when the node density reaches a high value, e.g., ten. In addition, Algorithm 2 allows concurrent transmissions when doing so reduces the overall transmission completion. Thus, there is not any negative $\Delta^-$ value for Algorithm 2. The $\Delta^-$ of Algorithm 2 fluctuates between 0.001% to 8%.

With increasing node density, from Figure 3.10, we observe that Algorithm 1 has a faster run time than others at approximately 0.02 ms initially and increases to 12 ms when the node density is fifteen. The reason is that it has the lowest run
3.5. Results

Figure 3.10: Computation time versus node density.

The run time complexity of $O(|AP||L|^2)$. Algorithm-2 has higher run time complexity and thus its run time increases faster than Algorithm-1. The run time of Algorithm-2 is about 0.11 ms initially and increases to about 40 ms when the node density is fifteen. The reason is because Algorithm-2 has a higher run time complexity of $O(|AP||L|^3)$. Algorithm-3 has the worst run time complexity, i.e., $O((|AP| + |Ω|)|L|^3))$, where its run time is recorded to be at 0.10 ms initially but increases to 256 ms when the node density is fifteen. These results confirm the theoretical analysis in Section 3.3.4.

3.5.2 Transmission Power

This experiment studies the impact of different transmission powers. The transmission power varies from 1 to 25 mW, with an interval of 1 mW. There are five APs and 25 clients. The SINR threshold for Algorithm-1 and Algorithm-SDT is 4 dB.

From Figure 3.11, 3.12 and 3.13, we observe that the $\Delta$ and $\Delta^+$ value of all algorithms have the same trend. The reason is that a higher transmission power means clients experience a stronger received signal. Consequently, all links are able to use a higher data rate. However, when the transmission power continues to increase, the interference between links also increases. Therefore, the value of $\Delta$ and $\Delta^+$ has
a decreasing trend after 2mW. The $\Delta$ of Algorithm 1, Algorithm 2, Algorithm 3, and Algorithm-SDT starts from 25%, 17%, 28% and 19%, respectively. After that, all algorithms experience a significant jump in their $\Delta$ value. In particular, the $\Delta$ value of Algorithm-1, Algorithm-2, Algorithm-3 and Algorithm-SDT reaches 45%, 39%, 47% and 41%, respectively.

![Figure 3.11: Average completion time reduction versus transmission power.](image)

In Figure 3.12, it can be observed that the $\Delta^+$ value of Algorithm-1, Algorithm-2, Algorithm-3 and Algorithm-SDT starts from 43%, 43%, 40% and 43%, respectively. Then, the $\Delta^+$ value of Algorithm-1, Algorithm-2, Algorithm-3 and Algorithm-SDT respectively reaches 61%, 62%, 66% and 56% when the transmission power increased to 2 mW. After that, the $\Delta^+$ of all algorithms has a decreasing trend.

From Figure 3.13, it can be observed that the $\Delta^-$ value of all tested algorithms starts from $-23\%$, $0.06\%$, $13\%$ and $-29\%$, respectively. Then, the $\Delta^-$ value of Algorithm 1, Algorithm 2, Algorithm 3 and Algorithm-SDT increases to 26%, 15%, 30% and 26%, thanks to the increased in transmission power. The $\Delta^-$ value of Algorithm 2 is the worst among all algorithms. The reason is because Algorithm 2 uses the average SNR value of links as the SINR threshold. When the transmission power is high, the threshold chosen by Algorithm 2 can be high, which leads to fewer links being scheduled concurrently. This explains why Algorithm 2 has the worst
3.5. Results

3.5.3 SINR Threshold Beta

This experiment studies the impact of different SINR thresholds $\beta$ on completion time reduction. The number of APs is five and the number of clients is 25. The transmission power is 10 mW. The experiment only considers Algorithm 1 and Algorithm-SDT because the other two algorithms choose an SINR threshold by themselves. From Figure 3.14, it can be observed that the $\Delta$ value of Algorithm-1 is about 26% initially. The $\Delta$ value of Algorithm-SDT is about five percentage points lower than that of Algorithm 1. The $\Delta$ value of both algorithms decreases with the SINR threshold $\beta$. The difference in $\Delta$ value between Algorithm 1 and Algorithm-SDT also decreases. When the SINR threshold reaches 10 dB, the $\Delta$ value of both algorithms is the same, which is about 7.5%. The $\Delta$ value of both algorithm reduces to only 5% when the SINR threshold $\beta$ is 12 dB. From Figure 3.15, we also observe a similar situation in terms of the $\Delta^+$ value of Algorithm 1 and Algorithm-SDT. The $\Delta^+$ value of Algorithm 1 is about 47% and the $\Delta^+$ of Algorithm-SDT is about five percentage points lower. The $\Delta^+$ of both algorithm reduces with SINR threshold $\beta$. 

Figure 3.12: Maximum completion time reduction versus transmission power.
3.5. Results

Figure 3.13: Minimum completion time reduction versus transmission power.

When $\beta$ is 12 dB, both algorithms have a $\Delta^+$ value of 10%. The reason is that when the SINR threshold $\beta$ increases, fewer links will be chosen to transmit concurrently due to higher interference. Algorithm 1 and Algorithm-SDT will only allow a small number of links to concurrently transmit when $\beta$ is high, e.g., 12 dB. Therefore, the performance of both algorithms decreases. The value of $\Delta^+$ also reduces to $\Delta$ because only a few links are allowed to transmit concurrently. However, the SINR threshold $\beta$ has no impact on the minimum number of concurrently transmitting links.

In Figure 3.16, the $\Delta^-$ of Algorithm 1 fluctuates between 2% to 8%. The $\Delta^-$ value of Algorithm-SDT fluctuates between $-11\%$ to 5%. When the SINR threshold $\beta$ is high, e.g., 10 dB, both algorithms only can find a small number of links that can transmit concurrently. Thus, the schedule obtained by both algorithms is similar to each other. The difference in $\Delta^-$ value between Algorithm 1 and Algorithm-SDT also decreases.
3.5. Results

Figure 3.14: Average completion time reduction versus SINR threshold $\beta$.

Figure 3.15: Maximum completion time reduction versus SINR threshold $\beta$. 
3.6 Conclusion

This chapter has addressed an important problem in dense WLANs comprising of APs equipped with an IBFD radio: deriving a schedule that allows nodes to complete the transmission of a given set of packets in minimum time. It proposes three novel algorithms to maximize the number of concurrent transmissions and also to determine the best data rate for use by each transmitting link. The results indicate that the proposed algorithms are able to reduce completion time by up to 68%. Moreover, the results show that the proposed algorithms are superior to prior algorithms that schedule links on a slot-by-slot basis.

A key future work is to design distributed algorithms that allow APs to complete their transmissions without the help of a controller. Another immediate work is to consider random channel gains, which affect the level of interference over time. Hence, in Chapter 4, this thesis presents a distributed Q-learning based scheduling algorithm that allows nodes to decide when to transmit and determine a data rate for each transmission under varying channel condition.
A Distributed Q-Learning Based Link Scheduler

As argued in Chapter 2, no existing link schedulers consider time varying or random channel condition. In particular, past works assume the channel condition is either fixed, or estimated before transmissions. Varying channel gains is a challenging issue. In particular, due to small-scale fading, the SINR at a receiver is uncertain. Consequently, a transmitter needs to select an appropriate data rate or Modulation Coding Scheme (MCS) for a given SINR, which is now affected by both the set of transmitting links and channel condition. However, in past works, they assume the SINR of links is a function of other activated links, and do not consider varying channel condition. Thus, past solutions cannot adapt to random increase or decrease in SINR after links are scheduled to transmit simultaneously.

This chapter outlines a problem involving scheduling full-duplex links under varying channel condition in one-hop wireless networks. A key challenge is that each node must first learn the contention pattern of neighbouring nodes before deciding when to transmit. In half-duplex scenarios, nodes will simply remain silent if they overhear one node has reserved the channel. However, in full-duplex scenarios, nodes
have an opportunity to transmit and should contend for the channel, even after a node has decided to transmit. Hence, each node must learn whether it can transmit concurrently with neighbouring nodes that have already decided to transmit. Secondly, each node must learn the optimal data rate for each transmission. Note that the interference from each neighbouring node may be different, and channel condition varies.

Henceforth, this chapter makes the following contributions:

C1 It presents a distributed link scheduler that is based on the Q-Learning algorithm [31]. The proposed algorithm allows nodes to learn which links to activate given other transmitting links, and also select a data rate that is robust against uncertain channel gains. Critically, nodes do not need to carry out expensive channel estimation and only require information from its neighbors as opposed to all nodes. Moreover, they learn to use half-duplex transmissions when channel condition deteriorates.

C2 The simulation results show that the proposed Q-Learning based link scheduler helps nodes to achieve successful transmissions under severe varying channel condition, where the channel gain varies from its original value to 30% of its original value. In addition, it fully utilizes the full-duplex capability of nodes to improve the overall throughput by 50% as compared to conventional Carrier Sense Multiple Access (CSMA). The simulation results shown that the required training time for nodes to learn the optimal schedule increases linearly with the number of nodes.

This chapter is organized as follows. Section 4.1 presents the system model and problem, followed by the hierarchical reinforcement learning approach used to address the problem in Section 4.1.3. Section 4.2 outlines the proposed Q-Learning based scheduling algorithm. Section 4.3 outlines the evaluation methodology. The results are presented in Section 4.4. Section 4.5 concludes the chapter.
4.1 Preliminaries

4.1.1 Network Model

The notations are summarized in Table 4.2. The set of nodes are denoted as \( N = \{1, \cdots , |N|\} \). All nodes have the same transmission range \( \Upsilon \). The Euclidean distance between node \( x \) and node \( y \) is denoted as \( d_{xy} \), where \( x, y \in N \). Let \( \mathcal{L}^x \) be the set of links where node-\( x \) is the transmitter. Let the set \( \mathcal{N}_x = \{y \mid d_{xy} \leq \Upsilon, y \in N \} \) contains all neighbors of node-\( x \). The transmission power of node-\( x \) is denoted as \( P_x \). All nodes have an IBFD radio with perfect self-interference cancellation. All nodes are located within the sensing range of each other. Let \( l_{xy} \) represent a directed link from node-\( x \) to node-\( y \). The random channel coefficient of link \( l_{xy} \) is denoted as \( g_{xy} \), which is calculated as follows,

\[
g_{xy} = \frac{c_0 F_g}{d^\omega} \quad (4.1)
\]

with \( c_0 = d_0^\omega 10^{-\frac{L_0}{10}} \) and \( F_g = 10^{-\frac{X_g}{10}} \). Here, \( \omega \) is the path loss exponent, \( L_0 \) is the path loss at reference distance \( d_0 = 1 \) meter, and \( X_g \) (in dB) is drawn from a Gaussian distribution with zero mean and standard deviation \( \sigma \).

The interference from link \( l_{xy} \) to link \( l_{uw} \) that is defined as,

\[
\iota_{xw} = P_x g_{xw} \quad (4.2)
\]

where \( P_x \) is the transmit power of node-\( x \), and \( g_{xw} \) is the path loss between node-\( x \) and node-\( w \).

The SINR \( \eta_{xy} \) of a link \( l_{xy} \) is defined as,

\[
\eta_{xy} = \frac{P_x g_{xy}}{\bar{I}_y + N_o} \quad (4.3)
\]

where \( \bar{I}_y = \sum_{u \in \mathcal{N}_y} \iota_{uy} \) is the sum interference at receiver \( y \), and \( N_o \) is the noise power. Given a SINR value, we can use Table 4.1 to obtain its corresponding data.
4.1. Preliminaries

rate. The set \( \Lambda \) denotes all the data rates shown in Table-4.1.

Table 4.1: SINR thresholds and their corresponding data rate [1]

<table>
<thead>
<tr>
<th>Thresholds (dB)</th>
<th>Data Rate ( \lambda ) (Mbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( 4 \leq \eta_{xy} &lt; 6 )</td>
<td>6</td>
</tr>
<tr>
<td>( 6 \leq \eta_{xy} &lt; 8 )</td>
<td>9</td>
</tr>
<tr>
<td>( 8 \leq \eta_{xy} &lt; 10 )</td>
<td>12</td>
</tr>
<tr>
<td>( 10 \leq \eta_{xy} &lt; 12 )</td>
<td>18</td>
</tr>
<tr>
<td>( 12 \leq \eta_{xy} &lt; 16 )</td>
<td>24</td>
</tr>
<tr>
<td>( 16 \leq \eta_{xy} &lt; 20 )</td>
<td>36</td>
</tr>
<tr>
<td>( 20 \leq \eta_{xy} &lt; 21 )</td>
<td>48</td>
</tr>
<tr>
<td>( \eta_{xy} \geq 21 )</td>
<td>54</td>
</tr>
</tbody>
</table>

Let \( \beta(\lambda) \) denote the minimum SINR value required to sustain data rate \( \lambda \). For example, referring to Table 3.1, the \( \beta(12) \) equals to 8 dB. The start and end time of a transmission over link \( l_{xy} \) and its duration is denoted as \( t_s(l_{xy}) \), \( t_e(l_{xy}) \) and \( \Delta(l_{xy}) = t_e(l_{xy}) - t_s(l_{xy}) \), respectively.

**Definition 3.** A transmission on link \( l_{xy} \) is successful if and only if the condition \( \eta_{xy} \geq \beta(\lambda) \) is true during \( \Delta_{xy} \).

Table 4.2: Symbols and Description

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N )</td>
<td>Set of nodes.</td>
</tr>
<tr>
<td>( l_{xy} )</td>
<td>A directed link between node-( x ) and node-( y ).</td>
</tr>
<tr>
<td>( d_{xy} )</td>
<td>The distance between node-( x ) and node-( y ).</td>
</tr>
<tr>
<td>( P_x )</td>
<td>Transmission power of node-( x ).</td>
</tr>
<tr>
<td>( G_r^x )</td>
<td>Receive antenna gain of node-( x ).</td>
</tr>
<tr>
<td>( G_t^x )</td>
<td>Antenna gain of node-( x ).</td>
</tr>
<tr>
<td>( N_x )</td>
<td>The neighbors of node-( x ).</td>
</tr>
<tr>
<td>( g_{xy} )</td>
<td>Channel gain for the directed link ( l_{xy} ).</td>
</tr>
<tr>
<td>( t_{xy} )</td>
<td>The interference from node-( x ) to node-( y ).</td>
</tr>
<tr>
<td>( \eta_{xy} )</td>
<td>SINR of of link ( l_{xy} ).</td>
</tr>
<tr>
<td>( t_s(l_{xy}) )</td>
<td>Start time of link ( l_{xy} ).</td>
</tr>
<tr>
<td>( t_e(l_{xy}) )</td>
<td>The end time of link ( l_{xy} ).</td>
</tr>
<tr>
<td>( \Delta(l_{xy}) )</td>
<td>Active duration of link ( l_{xy} ).</td>
</tr>
</tbody>
</table>
4.1.2 Problem Definition

Let $\Gamma^\tau$ and $\lambda^t$ be respectively the set of links and their corresponding data rate at time slot $\tau$. Also, $R^\tau(\Gamma^\tau, \lambda^t)$ is the sum rate of active links in the set $I^\tau$ that satisfy Definition-3. Define the random vector $g^t$ of dimension $|\mathcal{N}| \times (|\mathcal{N}| - 1)$ to contain channel coefficients at time $t$. The problem at hand is to select the set $\Gamma^\tau$ and $\lambda^t$ to maximize the following quantity:

$$R = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_{g^t} \left[ \sum_{t=0}^{T-1} R^\tau(\Gamma^\tau, \lambda^t) \right]$$ (4.4)

The aforementioned expectation is taken with respect to the joint probability of channel coefficients to nodes.

4.1.3 A Markov Decision Process Model

This section first presents a brief background on reinforcement learning and Hierarchical Reinforcement Learning (HRL) [132] before showing how the HRL framework is used to solve (4.4).

4.1.3.1 Background

Reinforcement learning is a machine learning technique where so called agents interact with the environment and learn the optimal strategy that maximizes a given reward [110][111]. The general process is shown in Figure 4.1(a). Let $\mathcal{S}$ be the set of states. First, an agent ascertains the current state $s$. Then, the agent has to choose and execute an action $a$ from its set of available actions $\mathcal{A}(s)$. After that it receives a reward/payoff $R(s,a)$. The environment or state then transitions to state $s'$. Let $P(s,a,s')$ denote the transition probability from state $s$ to $s'$ after taking action $a$. The agent employs a policy $\pi$ that specifies the action to be taken in each state. In addition, the agent maintains a so called value function $V^\pi$ that returns the expected overall reward using policy $\pi$. Specifically, given an infinite-horizon,
the discounted reward given state $s$ and following policy $\pi$ thereafter is defined as,

$$V^\pi(s) = \mathbb{E}\{R(s, \pi(s)) + \gamma R(s', \pi(s')) + \gamma^2 R(s'', \pi(s'')) \cdots | s\}. \quad (4.5)$$

Here, the symbol $\gamma$ denotes the discount, where $0 < \gamma < 1$. In words, future rewards are weighted less as compared to the current reward. Moreover, as the $\gamma$ is less then one, then $V^\pi(s)$ is guaranteed to converge over time. One way to compute the optimal value of $V^\pi$, denoted as $V^*$, for a given policy $\pi$, is via the following Bellman equation [110],

$$V^\pi(s) = \max_{a \in A(s)} \left[ R(s, a) + \gamma \sum_{s' \in S} P(s, \pi(s), s') V^\pi(s') \right] \quad (4.6)$$

A key assumption when computing (4.6) is that the transition probability between states is available. However, in practice, the transition probability $P(s, \pi(s), s')$ is unknown. To this end, the problem requires a model-free approach. In particular, one can employ Q-Learning [133]. The agent maintains a Q-factor $Q(s, a)$ for each state $s$ and action $a$. Specifically, at the $\tau$-th iteration or time slot and given step size $\alpha$, the Q-factor is calculated as,

$$Q^{\tau+1}(s, a) = (1 - \alpha)Q^\tau(s, a) + \alpha \left[ R(s, a) + \gamma^\tau \max_{a' \in A(s')} Q^\tau(s', a') \right] \quad (4.7)$$

As proved in [133], if each state $s \in S$ is visited infinitely often, then an agent will find the optimal value of each Q-factor $Q(s, a)$. An agent can then simply determine the optimal action for state $i$ by calculating $\arg \max_{a \in A(s)} Q(s, a)$.

Reinforcement learning has a key limitation [132]: it assumes time is discrete and the transition time between states is negligible. However, in some problem instances such as the one considered in this chapter, in each transition, there may be one or more sub-tasks, which may take a variable number of time slots to complete. More-
4.1. Preliminaries

Figure 4.1: Reinforcement learning versus hierarchical reinforcement learning.

over, each sub-task contains at least one sub-action. The problem in this chapter for example, an agent has the task of selecting a link followed by another task, which is to select a suitable data rate. These sub-tasks have a variable finishing time as the chosen data rate will dictate when a transmission completes. Alternatively, an agent may decide not to transmit.

The aforementioned limitation is addressed by Hierarchical Reinforcement Learning (HRL) [134]. Referring to Figure 4.1(b), an agent has one or more sub-tasks; each of which takes variable number of time slots to complete. Each sub-task may be further decomposed into other sub-tasks. Each sub-task has its own actions, goal and policy. Define \( M = \{ m_1, m_2, \ldots, m_M \} \) as the set of sub-tasks. Each sub-task \( m^k \) has a sub-action \( a_{m^k} \) and a sub-policy \( \pi_{m^k} \). Each sub-action \( a_{m^k} \) if taken generates a sub-reward \( R(s, a_{m^k}) \). An action \( a \in A(s) \) consists of sub-actions \( a = \{ a_{m^1}, a_{m^2}, \ldots, a_{m^M} \} \); see Figure 4.1(b). Once all sub-actions of \( a \) have been executed, state \( s \) will transition to the next state \( s' \). The time for state \( s \) to transition to the next state \( s' \) is called the transition duration and it is denoted as \( t(s, a, s') \).

The reward \( R(s, a) \) for action \( a \) is the sum of all sub-reward \( R(s, a_{m^k}) \) divided by transition duration \( t(s, a, s') \). Formally, the reward is calculated as,

\[
R(s, a) = \frac{\sum_{k=1}^{M} R(s, a_{m^k})}{t(s, a, s')} \tag{4.8}
\]

The sub-policy \( \pi_k(s) \) indicates the sub-action \( a_{m^k} \) chosen for sub-task \( m^k \) under state \( s \). The optimal policy \( \pi^* \) is now a set of optimal sub-policy \( \pi_{m^k}^* \) that maximizes (4.8)
for every state visited by the agent. In addition, the optimal policy $\pi^*$ is defined as the set $\{\pi^*_m, \pi^*_m, \ldots, \pi^*_m|_{M_i}\}$, where $\pi^*_m(i)$ returns the optimal sub-action $a_{m^k}$ for sub-task $m^k$ in state $i$.

As agents employ Q-Learning, they have to maintain a Q-factor for each sub-action. At the $\tau$-th iteration and given step size $\alpha$, the Q-factor is calculated as,

$$Q^{\tau+1}(s, a_{m^k}) = (1 - \alpha)Q^{\tau}(s, a_{m^k}) + \alpha \left[ R(s, a_{m^k}) + \gamma^\tau \max_{a'_{m^k} \in A_{m^k}(s')} Q^{\tau}(s', a'_{m^k}) \right]$$

Assuming an agent visits each state in $S$ infinitely often, then the $Q(s, a_{m^k})$ value for each sub-action will converge to the optimal value. The agent can determine the optimal action $a$ by determining all optimal sub-action $a_{m^k}$ by calculating $\arg \max_{a_{m^k} \in A_{m^k}(s)} Q(s, a_{m^k})$. The R-SMART algorithm [135] is used to calculate Q-factors. Specifically, an agent maintains an average reward defined as,

$$\rho^{\tau+1} = (1 - \beta) \times \rho^{\tau} + \beta \times \frac{\sum_{s \in S} R(s, a)}{\sum_{s \in S} t(s, a, s')}$$

The Q-factor $Q(s, a)$ is then calculated as,

$$Q^{\tau+1}(s, a) = (1 - \alpha)Q^{\tau}(s, a) + \alpha \left[ R(s, a) - \rho^\tau t(s, a, s') + \gamma^\tau \max_{a' \in A(s')} Q^\tau(s', a') \right]$$

The value of $\alpha$ and $\beta$ in the R-SMART algorithm converges to zero when the number of iterations $|\tau|$ approaches infinity.

### 4.1.4 A HRL Model

This chapter are now ready to present an HRL model of the problem. This model captures a node learning the most suitable link to be activated and the highest data
4.1. Preliminaries

Figure 4.2: A hierarchy of actions.

rate that can be supported by the link. Next, this section makes specific the state, action, reward, transition duration and objective, of the HRL model:

1. **State.** The system state is \( s^x = \{I(y) \mid y \in \mathcal{N}_x\} \), where the function \( I(y) \) is an indicator function that returns either a zero or one. It returns a value of one if a neighbour \( y \) has chosen to transmit; this also means its data rate is non-zero. Otherwise, if \( I(y) \) returns zero, then node-\( y \) is silent, meaning node-y has a data rate of zero. As an example, reconsider Figure 1.1. Node-1 has three neighbors: node-2, node-3, and node-4. If node-1 has the system state \( s^1 = (0, 1, 0) \), then it indicates node-2 and node-4 have chosen not to transmit. On the other hand, node-3 has chosen to transmit.

2. **Action.** For a given node-\( x \), as shown in Figure 4.2, its action \( a^x = \{l, \lambda\} \) is a set of two sub-actions: (i) \( l \), which refers to a chosen link, and (ii) \( \lambda \), which refers to a data rate from Table 3.1. The action space of node-\( x \) is \( \mathcal{A}^x(s) = \mathcal{L}^x \times \Lambda \). For example, node-1 in Figure 1.1 has links \( l_{12} \) and \( l_{13} \). Hence, \( \mathcal{L}^1 = \{l_{12}, l_{13}\} \). As per Table-3.1, we have \( \Lambda = \{0, 6, 9, 12, 18, 24, 36, 48, 54\} \). Therefore, an action of node-1 is \( a^1 = \{l_{12}, 6\} \), which means node-1 chooses to activate link \( l_{12} \) at 6 Mbps.

3. **Reward.** Define \( R(s, a) \) as the reward for a node executing action \( a \) under system state \( s \). The reward \( R(s, a) \) is equal to sub-reward \( r(s, l) \) plus sub-reward \( r(s, \lambda) \). The sub-reward \( r(s, l) \) is obtained when a node selects link \( l \) under
4.2 Q-Learning Based Link Scheduling Algorithm

state \( s \), whereas sub-reward \( r(s, \lambda) \) corresponds to the payoff for choosing data rate \( \lambda \) under state \( s \). The exact definition of sub-rewards \( r(s, l) \) and \( r(s, \lambda) \) will be presented in Section 4.2.

4. Transition Duration. Let \( N^s_x \) be a set that contains node-\( x \)'s neighbours that choose to transmit when node-\( x \) detects state \( s^x \). The nodes included in the set \( \{ N^s_x, x \} \) have chosen a link to activate. Then, the transition duration \( t(s, a^x, s') \) is calculated as,

\[
t(s, a^x, s') = \max \Delta(l)
\]  

where \( l \in a^x \) and \( y \in \{ N^s_x, x \} \). For example, reconsider Figure 1.1. Assume node-1 has the system state \( s^1 = (0,1,0) \), which means node-3 will activate link \( l_{31} \). In addition, assume node-1 chooses action \( a^1 = \{ l_{12}, 6 \} \), meaning node-1 will activate link \( l_{12} \). If the condition \( \Delta(l_{12}) > \Delta(l_{31}) \) is true, then the transition duration is \( t(s, a^1, s') = \Delta(l_{12}) \). Otherwise, the transition duration is \( t(s, a^1, s') = \Delta(l_{31}) \).

Next, this chapter presents the Q-Learning based link scheduler to solve the foregone HRL model.

4.2 Q-Learning Based Link Scheduling Algorithm

Algorithm-4 shows the steps run by a node to activate a link and to select a data rate. The node starts with the standard random back-off. Then, the node calls function \texttt{Negotiation}() to determine the correct state; see lines 4 - 5. This section first motivates the reason nodes carry out multiple message exchanges to find the correct state. Consider the following scenario. Assume node-\( x \) and node-\( y \) inform each other of their chosen action only \textit{once}. That is, node-\( x \) informs node-\( y \) that it is planning to transmit, and vice-versa. After that, assume node-\( x \) decides not to transmit after receiving node-\( y \)'s action as it deems doing so will yield a low reward.
4.2. Q-Learning Based Link Scheduling Algorithm

However, node-\(y\) decides to transmit and its transmission is successful. Observe that node-\(y\) has an inconsistent state where it concludes incorrectly that its transmission is successful when it transmits simultaneously with node-\(x\). However, node-\(x\) did not transmit in this case. This example shows that both node \(x\) and \(y\) need multiple message exchanges before they have the same state information. Otherwise, their action and reward will be attributed to an incorrect state.

To carry the negotiation, nodes exchange a special control frame called *Negotiation to Send* (NTS). The NTS frame contains the state and chosen action of the node, denoted as \(\text{NTS}\{\text{state, flag, token}\}\). The flag is equal to one if the node has not visited the state recorded in the NTS frame. Otherwise, the flag is zero. To avoid collision, nodes transmit a NTS frame in turns. Specifically, each NTS frame contains a token field, which is set to the ID of the next node to transmit a NTS frame. A node that sees its ID in the token field of the NTS frame then proceeds to transmit a NTS frame. Figure 4.3 shows an example where nodes send a NTS frame with the token field set to the next transmitting node.

![Figure 4.3: Three nodes transmit an NTS frame according to their ID.](image)

Nodes call the function \texttt{Negotiation(.)} to begin the negotiation stage. There are two cases to consider. The \textit{first} case is where a node completes its random back-off, see line 1 to 18 in Algorithm 5. The \textit{second} case is where the node is still in the back-off stage but detects the channel is busy, see line 20 to 35 of Algorithm 5.
4.2. Q-Learning Based Link Scheduling Algorithm

In the first case, after a node’s back-off expires, the node initializes its state as \( \{0, \cdots, 0\} \). Then, the node sets itself to timeout in \( t_{NTS} \) seconds.

It then executes the following steps:

1. Choose an action based on the current state by calling the function \texttt{ChooseAction(.).} The function \texttt{ChooseAction(.)} returns an action for the current state, and also a status that indicates whether the node has visited the current state. The details of \texttt{ChooseAction(.)} will be elaborated later.

2. If the node has visited the current state, it transmits an NTS frame with \( \text{flag} = 0 \), the current state and chosen action. The token is set to the next in order ID. If the node has not visited the current state, in addition to setting \( \text{flag} = 1 \), it resets its timeout to \( t_{NTS} \) seconds.

3. The node waits to receive a NTS frame. It then updates its current state based on the information carried by the received NTS frame. If the NTS frame indicates that the neighbor chooses a non-zero data rate as an action, then the node considers the neighbor is going to transmit. Otherwise, its neighbor is marked as idle. If the last received NTS frame has \( \text{flag} = 1 \), then it extends the negotiation stage by restarting the timeout for \( t_{NTS} \) seconds. If the token does not equal the ID of the node, then it re-executes the step 3). Otherwise, it goes back to step 1), assuming there is no timeout. If there is a timeout, the node ends its negotiation.

In the second case, once a node detects that the channel is busy while in back-off, it immediately terminates its back-off. Then, the node then initializes its state as empty, and sets a timeout of duration \( t_{NTS} \) seconds. It then executes the following steps:

1. It awaits a NTS frame from a neighbor. It then refreshes the current state based on the information carried by the NTS frame. If the NTS frame sent by the neighbor indicates that the neighbor chooses a non-zero data rate, then
the node concludes that the neighbor is going to transmit. Otherwise, it marks
the neighbor as idle.

2. If the flag of the received NTS frame is one, it resets its timeout to \( t_{\text{NTS}} \). If the
token field does not contain its ID, it goes back to the first step. Otherwise, it
chooses an action by calling the function \texttt{ChooseAction(\)}. If the node has
visited the current state, it transmits a NTS frame with \( \text{flag} = 0 \), its current
state and chosen action. The token is set to the ID of the next node. If the
node has not visited the current state, in addition to setting \( \text{flag} = 1 \), it resets
its timeout to \( t_{\text{NTS}} \). If there is no timeout, the node goes back to step one.
Otherwise, the node ends the negotiation.

After negotiation, nodes record the state and the last chosen action and enter
the transmit stage, see line-7 in Algorithm 4. They start to transmit according
to their chosen action. Then, all nodes wait for an ACK packet. Upon receiving
an ACK packet, they move into the update stage to receive a reward and update
the corresponding Q-factor. In particular, nodes obtain reward \( r(s^x, \lambda) \) and \( r(s^x, l) \),
for their chosen data rate and link, respectively; see Algorithm 6 and 8. Finally,
nodes update Q-factor \( Q(s^x, \lambda) \) and \( Q(s^x, l) \) using the function \texttt{Qupdate1(\)} and
\texttt{Qupdate2(\)}; see Algorithm 7 and 9.

**Algorithm 4: A Q-Learning Based Scheduling Algorithm**

1. \( T = 0; \)
2. \( R_{\lambda} = 0; \)
3. \( R_l = 0; \)
4. Start random back-off;
5. \([s^x, a^x] = \text{Negotiation(\)};\)
6. Record state \( s^x \) and action \( a^x = \{l, \lambda\}; \)
7. Transmit on link \( l \) with data rate \( \lambda; \)
8. Wait for an ACK packet;
9. Receive reward \( r(s^x, \lambda); \)
10. \textbf{Call} \texttt{Qupdate1}(s^x, \lambda, r(s^x, \lambda), T, R_{\lambda}); \)
11. Receive reward \( r(s^x, l); \)
12. \textbf{Call} \texttt{Qupdate2}(s^x, l, r(s^x, l), T, R_l); \)
13. Go to line 4;
Algorithm 5: Negotiation(.)

1. if back-off expires then
2.     \( s^x = \{0, \ldots, 0\} \);
3.     SetTimeout(\( t_{NTS} \));
4.     while not timeout do
5.         \([a^x, \text{status}] = \text{ChooseAction}(s^x)\);
6.         if \text{status}='StateIsNew' then
7.             Send an NTS frame with flag=1, token=x.ID+1;
8.             SetTimeout(\( t_{NTS} \));
9.         else
10.            Send an NTS frame with flag=0, token=x.ID+1;
11.        end
12.        NTS = ReceiveMessage();
13.        Update state \( s^x \);
14.        if \( NTS.\text{flag}==1 \) then
15.            SetTimeout(\( t_{NTS} \));
16.        end
17.        if \( NTS.\text{token} == x.ID \) then
18.            Go to line 5;
19.        else
20.            Go to line 11;
21.        end
22.    end
23. else
24.    if channel becomes busy then
25.        terminate back-off;
26.        \( s^x = \emptyset \);
27.        SetTimeout(\( t_{NTS} \));
28.        while not timeout do
29.            NTS = ReceiveMessage();
30.            Update state \( s^x \);
31.            if \( NTS.\text{flag}==1 \) then
32.                SetTimeout(\( t_{NTS} \));
33.            end
34.            if \( NTS.\text{token} == x.ID \) then
35.                \([a^x, \text{status}] = \text{ChooseAction}(s^x)\);
36.                if \text{status}='StateIsNew' then
37.                    Send an NTS frame with flag=1, token=x.ID+1;
38.                    SetTimeout(\( t_{NTS} \));
39.                else
40.                    Send an NTS frame with flag=0, token=x.ID+1;
41.                end
42.            end
43.        end
44.    end
45. end
4.2. Q-Learning Based Link Scheduling Algorithm

The function ChooseAction(.) will firstly determine whether the node has visited the state \( s^x \) during the current negotiation. If the node has previously visited state \( s^x \), the function ChooseAction(.) will return the action selected when it first visited the said state. The variable status will return as StateIsOld. If a node has never visited the state \( s^x \) in the current negotiation, the function ChooseAction(.) sets the variable status as StateisNew. Then, nodes have to randomly choose an action for the new state. For achieving this, the function ChooseAction(.) assigns a probability to each action based on parameters \( G_1 \) and \( G_2 \). An action \( a \) is a greedy action if it has the largest Q-factor \( Q(s, a) \) among all actions \( A(s) \) that are available for the current state \( S^x \). The greedy action has probability \( p_g \) of being chosen. If all actions have the same Q-factor, then all actions are considered to be greedy actions. This probability \( p_g \) is calculated by,

\[
p_g = 1 - \left( \frac{G_1}{G_2 + \tau} \right) \times \left( \frac{1}{|A^x(i)| - 1} \times \frac{G_1}{G_2 + \tau} + (1 - \left( \frac{G_1}{G_2 + \tau} \right)) \right)
\]  

(4.13)

where \( \tau \) is the number of iterations. All other actions are considered to be non-greedy and have a probability \( p_{ng} \) of being chosen. The probability \( p_{ng} \) is calculated as,

\[
p_{ng} = \left( \frac{G_1}{G_2 + \tau} \right) \times \left( \frac{1}{|A^x(i)| - 1} \times \frac{G_1}{G_2 + \tau} + (1 - \left( \frac{G_1}{G_2 + \tau} \right)) \right)
\]  

(4.14)

In Equ. (4.13) and Equ. (4.14), the value of \( G_1/(G_2 + \tau) \) decreases with each iteration of the algorithm. On the contrary, the value of \( 1 - (G_1/(G_2 + \tau)) \) increases with the number of iterations. Hence, the probability that nodes will choose a favourable action will increase with the number of iterations. Nodes will explore new actions initially and gradually converge to the best action over time, which accelerates the convergence of Q-factors.

Algorithm-6 calculates the sub-reward \( r(s^x, \lambda) \). If the transmission is successful according to Definition-3, the reward \( r(s^x, \lambda) \) is set to \( \varphi \) times the chosen data rate. Otherwise, it is equal to minus \( \varphi \) times the chosen data rate. The parameter \( \varphi \) is
used to enlarge the differences between the reward of data rates.

**Algorithm 6:** $r(s^x, \lambda)$  

| **input:** | $s^x$, $\lambda$  
<table>
<thead>
<tr>
<th><strong>output:</strong></th>
<th>sub-reward $r(s^x, \lambda)$</th>
</tr>
</thead>
</table>
| 1 | if *Transmission is successful as per Definition-3* then  
| 2 | return $r(s^x, \lambda) = \varphi \times \lambda$;  
| 3 | else  
| 4 | return $r(s^x, \lambda) = -\varphi \times \lambda$;  
| 5 | end |

The function **Qupdate1** is shown by Algorithm-7 which updates the Q-factor $Q(s^x, \lambda)$. The overall transition duration is recorded by parameter $T$. The overall rewards from sub-actions $\lambda$ is recorded by parameter $R_{\lambda}$. Then, $\rho_{\lambda}$ is the average reward for sub-action $\lambda$. Q-factor $Q(s^x, \lambda)$ is updated in line 4 according to Equ. (4.11).

**Algorithm 7: Qupdate1**  

| **input:** | $s^x$, $\lambda$, $r(s^x, \lambda)$, $T, R_{\lambda}$  
<table>
<thead>
<tr>
<th><strong>output:</strong></th>
<th>$Q(\lambda, s^x)$</th>
</tr>
</thead>
</table>
| 1 | $T = T + t(s^x, a^x, s^{x'})$;  
| 2 | $R_{\lambda} = R_{\lambda} + r(s^x, \lambda)$;  
| 3 | $\rho_{\lambda} = (1 - \beta) \times \rho + (\beta \times \frac{R_{\lambda}}{T})$;  
| 4 | Calculate $Q(s^x, \lambda)$ as per (4.11) |

Algorithm-8 shows the calculation of sub-reward $r(s^x, l)$. It finds the largest Q-factor $Q(s^x, \lambda)$ and returns its value as the reward for choosing link $l$.

**Algorithm 8: r($s^x, l$)**  

| **input:** | $s^x$, $l$  
<table>
<thead>
<tr>
<th><strong>output:</strong></th>
<th>sub-reward $r(s^x, l)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$r(s^x, l) = \max_{\lambda \in \Lambda} Q(s^x, \lambda)$</td>
</tr>
</tbody>
</table>

The function **Qupdate2(.),** as outlined in Algorithm-9, updates the Q-factor $Q(s^x, l)$. Similar to **Qupdate1(.),** the overall reward from sub-action $l$ is recorded in $R_l$. The average reward for sub-action $l$ is denoted as $\rho_l$. The Q-factor $Q(s^x, l)$ is updated in line 4 according to Equ. (4.11).

This section concludes with the following proposition.
Algorithm 9: Qupdate2

\begin{itemize}
  \item \textbf{input} : $s^x$, $l$, $r(s^x, l), T, R_l$
  \item \textbf{output}: $Q(s^x, l)$
  \begin{enumerate}
    \item $T = T + t(s^x, a^x, s^x')$;
    \item $R_l = R_l + r(s^x, l)$;
    \item $\rho_l = (1 - \beta) \times \rho + \beta \times \frac{R_l}{T}$;
    \item Calculate $Q(S^x, \lambda)$ as per (4.11);
  \end{enumerate}
\end{itemize}

Proposition 6. The negotiation stage always ends.

Proof. This proof shows that all nodes will timeout; i.e., nodes will not reset their timeout to $t_{NTS}$ continuously. First, a node will not reset its timeout value if it has previously visited a state; see line 10 and 35 of Algorithm 5. Secondly, there are finite number of states. In particular, a node with $|N|$ neighbors will have $2^{(|N| - 1)}$ states. In the worst case, a node will iterate through all these states and assign each one an action. At such time, a node will no longer reset its timeout value, and thus ends the negotiation process as claimed. Note that in practice, the negotiation is short after training; see Section 4.4.3. \qed

4.3 Evaluation

The evaluation is conducted over a discrete-time event simulation in C# to validate the proposed approach. Each time step is 1 ms in length. All nodes are saturated, meaning they always have data packets to send. The evaluation uses the simulation parameters shown in Table 4.3. The parameters $G_1$ and $G_2$ are set to 100 and 200; these values ensure nodes adequately explore their action space to determine the best action for a given state. All NTS frames are assumed to be received by nodes successfully. In the experiments, $t_{NTS}$ is set to 5 ms to allow sufficient time for nodes to select an action and send a NTS frame. The experiments study the following parameters:

\begin{enumerate}
  \item Learning rate $\alpha$. As discussed in Section 4.1.3, the value of $\alpha$ influences the convergence speed of Q-factors. According to [135], the value of $\alpha$ must de-
crease with increasing number of iterations. To this end, the evaluation considers three different approaches to adjust $\alpha$. Denote the learning rate of each approach as $\alpha^1$, $\alpha^2$ and $\alpha^3$; how each one is calculated will be detailed in Section 4.4.1.

2. Discount factor $\gamma$. As shown in Section 4.1.3, the value of $\gamma$ also influences the value of Q-factors. The $\gamma$ is set to 0.99, 0.95, 0.90, 0.85, 0.80 or 0.75.

3. Number of nodes $|N|$, where $|N| \in \{2, 3, 4, 5\}$.

4. Channel condition. The severity of the channel is determined by the variance of the Gaussian distribution; see Equ.-(4.1). In the experiments, the values of $\sigma$ (in dB) are drawn from the range $[0, 5]$.

Table 4.3: Simulation parameters

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grid size</td>
<td>$25m \times 25m$</td>
</tr>
<tr>
<td>Transmit power $P_t$</td>
<td>100 mW</td>
</tr>
<tr>
<td>Antenna gain $G_r$ and $G_t$</td>
<td>2 dB</td>
</tr>
<tr>
<td>Time step</td>
<td>1 ms</td>
</tr>
<tr>
<td>Simulation duration</td>
<td>10 min</td>
</tr>
<tr>
<td>Packets length</td>
<td>20 to 65535 bytes</td>
</tr>
<tr>
<td>Discount factor $\gamma$</td>
<td>0.99, 0.95, 0.90, 0.85, 0.80 and 0.75</td>
</tr>
<tr>
<td>Learning rate $\beta$</td>
<td>$90/(100 +</td>
</tr>
<tr>
<td>$t_{NTS}$</td>
<td>5 ms</td>
</tr>
<tr>
<td>$t_{ACK}$</td>
<td>2 ms</td>
</tr>
<tr>
<td>$G_1$</td>
<td>100</td>
</tr>
<tr>
<td>$G_2$</td>
<td>200</td>
</tr>
<tr>
<td>$\varphi$</td>
<td>10</td>
</tr>
</tbody>
</table>

As part of the evaluation, nodes also use CSMA and TDMA. When nodes use CSMA, they will initially choose the lowest data rate from Table-3.1. For every five successful transmissions, a node will increase its data rate to the next higher data rate. Otherwise, its data rate is lowered to the next level. As for TDMA, each node is assigned with a time slot. The length of each time slot is set to $\frac{65535\times8}{6\times10^6} \approx 88$ ms. This ensures that each node will be able to send at least one data packet during its
own time slot. Nodes select a data rate using one of the following methods: TDMA\(^1\) and TDMA\(^2\). In TDMA\(^1\), nodes will choose the highest data rate that the channel condition can support when they start to transmit. In TDMA\(^2\), nodes will always choose 6 Mbps which is the lowest data rate in Table 3.1.

Each simulation is divided into two stages. First, there is the *training* stage, which is used to obtain the optimal Q-factors. In the training stage, nodes transmit 20 bytes data packets. Packet lengths are drawn randomly from the range [20, 65536] (in bytes) After that, there is the *transmit* stage, which is divided into two sub-stages. The goal is to compare the performance of CSMA, TDMA\(^1\) and TDMA\(^2\) against the proposed approach using the same setup. Nodes use their Q-factors to select links and data rates for transmission in the first 10 minutes. In the second 10 minutes, nodes employ CSMA. After that, they use TDMA\(^1\) for 10 minutes. In the last ten minutes, nodes use TDMA\(^2\). The following metrics are collected: (i) *Training time*. This is the time that is required for all nodes to learn the optimal transmit policy, (ii) *Number of NTS frames*. This corresponds to the number of NTS frames sent before each transmission, which quantifies the signaling overheads of the negotiation stage, and (iii) *Average throughput*. This is the average throughput of the first, second, third and fourth 10 minutes of the transmit stage for the approach, CSMA, TDMA\(^1\) and TDMA\(^2\), respectively.

### 4.4 Results

The next section studies the impact of learning rate \(\alpha\). The following section considers the discount factor \(\gamma\). Section 4.4.3 investigates node density followed by different channel condition.

#### 4.4.1 Learning Rate

The topology shown in Figure 4.4 is used to study the impact of the learning rate \(\alpha\) on the average throughput. The discount factor \(\gamma\) is set to 0.99 and \(\sigma\) is set to 1
4.4. Results

dB.

![Topology with five nodes.]

Figure 4.4: Topology with five nodes.

Figure 4.5 shows the evolution of $\alpha^1, \alpha^2, \alpha^3$. The value of $\alpha^1$ is set to $\frac{\log(\tau+1)}{\tau+1}$, where $\tau$ is the number of iterations. The value of $\alpha^1$ decreases fastest at the beginning before it decreases slower than $\alpha^2$ and $\alpha^3$ with increasing number of iterations. The value of $\alpha^2$ is calculated as follows,

$$\alpha^2 = \frac{10^{100 \times G_1 - G_2} - 10^{\tau - 1}}{10^{100 \times G_1 - G_2} \times 2.73}$$  \hspace{1cm} (4.15)$$

The value of $\alpha^2$ decreases at the slowest rate. Note that the value of $\alpha^2$ does not converge to zero because the maximum number of iterations of the training stage is $100 \times G_1 - G_2$. During the training stage, after $100 \times G_1 - G_2$ iterations, the probability that nodes explore non-greedy actions is less than 0.01, see the function `ChooseAction(.)` in Section 4.2. The $\alpha^3$ is calculated as,

$$\alpha^3 = 0.3662 \times \frac{100 \times G_1 - G_2}{G_1 - G_2} \times 10^{\tau - 1} - (\tau - 1)$$  \hspace{1cm} (4.16)$$

Figure 4.6 shows the average throughput for different learning rates. For each learning rate, the simulation is run for ten times to collect the average result. The average throughput for $\alpha_1$ and $\alpha_2$ differs only by 1.1 Mbps. For $\alpha_3$, the average throughput is 23.9 Mbps. As per Algorithm 8 and 9, for high learning rates, nodes will use the immediate reward to update their Q-factors as opposed to rewards received in previous iterations. As per Figure 4.5, we see that in order to achieve the
4.4. Results

Figure 4.5: Number of iterations versus the values of learning rate $\alpha$.

best result, the immediate reward should be weighted less as compared to historical rewards. This is reasonable as random channel condition mean the current channel gain may not reflect the long term trend. For $\alpha^2$, the average throughput also decreases. This is because $\alpha^2$ has a higher value than $\alpha^1$, causing nodes to use the immediate reward when updating their Q-factors. Thus, the average throughput decreases. Hence, all subsequent experiments will use $\alpha_1$ as the learning rate.

Figure 4.6: Learning rate $\alpha^1$, $\alpha^2$ and $\alpha^3$ versus the number of iterations.
4.4.2 Discount Factor

This experiment uses the same topology as in Section 4.4.1, the learning rate is $\alpha^1$, and $\sigma$ is set to 1 dB. From Figure 4.7, we observe that $\gamma$ does not affect the average throughput. The reason is that the next state of nodes is not only determined by their current action but also the next negotiation. In addition, a node’s current transmission is independent of the node’s future transmissions. If nodes discover an action yields a high reward under their current state, they must always receive a similar reward when the same state re-occurs. Hence, nodes will always learn the same strategy no matter they are focusing on the current or future reward. As a result, the discount factor $\gamma$ does not influence the average throughput.

Figure 4.8 shows the cumulative average reward for choosing a data rate. The cumulative average reward is defined as the ratio between the total reward for choosing a data rate and the number of iterations thus far. For different $\gamma$ values, the cumulative average reward fluctuates initially. After that, it converges to around 3800 under all different values of $\gamma$. When using different values of $\gamma$, the speed of convergence is similar. When using $\gamma = 0.75$, the speed of convergence is appreciably faster over time. The reason is that a lower $\gamma$ value means there is less change in Q-factor values. In other words, the value of Q-factor converges faster. When Q-factors have converged, the cumulative average reward also converges. Hence, subsequent experiments use $\gamma = 0.75$ because it does not influence the optimal strategy while having a slightly faster convergence speed.

4.4.3 Number of Nodes

In this experiment, the topologies are shown in Figure 4.9, which correspond to different node densities. This experiment uses the following parameter value: $\sigma = 1$ dB, $\gamma = 0.75$, and $\alpha_1$.

In Figure 4.10, we observe that the average throughput only changes slightly with increasing number of nodes. Advantageously, the average throughput achieved
4.4. Results

Figure 4.7: Discount factor $\gamma$ versus the average throughput.

Figure 4.8: Cumulative average reward for choosing a data rate versus the number of iterations for different discount factor $\gamma$. 
4.4. Results

Two Nodes Topology

Three Nodes Topology

Four Nodes Topology

Five Nodes Topology

Figure 4.9: Topologies with different number of nodes.

by the proposed link scheduler is twice that of CSMA because nodes learn that a relay or bi-directional full duplex transmission is the optimal strategy in every time slot. They also learn the existence of a third concurrent transmission causes too much interference, and thus do not transmit. Hence, there will always be two concurrent transmissions in every time slot. In addition, nodes will use the highest data rate that leads to the highest number of successful transmissions. When nodes use CSMA, there will only be a half-duplex transmission in every time slot. Hence, the throughput achieved by the proposed Q-Learning algorithm is double that of CSMA. When nodes use TDMA\textsuperscript{1} or TDMA\textsuperscript{2}, nodes transmit one after another. However, for TDMA\textsuperscript{1}, nodes select the highest possible data rate supported by the channel condition when the transmission starts. During data transmission, the selected data rate may not be suitable for the channel condition. Therefore, when nodes use TDMA\textsuperscript{1}, their transmissions are likely to fail. As a result, the average throughput achieved by TDMA\textsuperscript{1} is lower than CSMA. As for TDMA\textsuperscript{2}, nodes always use the lowest data rate. Hence, the average throughput achieved by TDMA\textsuperscript{2} is the lowest.
4.4. Results

Figure 4.10: Average throughput over different topologies.

Figure 4.11 shows how many NTS frames with flag= 1 have been sent in the first 4000 iterations of the training stage. All node must send at least one NTS frame with flag= 1 in every negotiation. Hence, the number of NTS frames with flag= 1 in each negotiation must be at least $|N|$, where $|N|$ is the number of nodes. Each node must have a maximum of $2^{(|N|−1)}$ possible states. Thus, the number of NTS frames with flag= 1 in each negotiation does not exceed $|N| \times 2^{(|N|−1)}$ during the negotiation process; this is confirmed in Figure 4.11. With each new node, the number of NTS frames with flag= 1 will increase at least by one in every negotiation. Hence, nodes have to reset the timeout at least one more time. As a result, the negotiation time increases in proportion to the number of nodes. Note that the average throughput, however, does not decrease significantly. In particular, the average throughput decreases by about 0.2 Mbps with each new node. This is because the negotiation duration remains short.

Figure 4.12 shows the average number of NTS frames with flag= 1 that are sent after nodes are trained. The average number of NTS frames increases from three when the number of nodes is two, to 6.903 when the number of nodes is five. However, this increase becomes slower with the number of nodes. This is because after training nodes learned that there can only be two concurrent transmissions. As shown in Figure 4.3, nodes send NTS frames sequentially. The first node that
4.4. Results

finishes random back off and starts the negotiation must choose to transmit because its initial state is \( \{0, 0, \cdots, 0\} \) which means no node chooses to transmit. The first node must visit two states in every negotiation: 1) its initial state, and 2) another node choosing to transmit. The remaining nodes will possibly visit two states: 1) the first node choosing to transmit and no other node choosing to transmit; 2) the first node and another node choosing to transmit. Hence, the minimum number of NTS frames with flag= 1 is \( |N| + 1 \) which happens when the first node visits two states and the rest nodes only visit one state. The maximum number of NTS frames with flag= 1 is \( 2 \times |N| \) which happens when all nodes visit two states. Compared to the training stage, the maximum number of NTS frames with flag= 1 reduces significantly. Hence, the negotiation duration remains short regardless of the number of nodes after training.

Figure 4.11: Number of NTS frames with \( flag = 1 \) sent in the training stage.

Figure 4.13 shows that the required training time increases with the number of nodes at a nearly constant rate. The reason is because if all nodes are within each other’s transmission range, adding a new node into the topology will create \( |N| \) new links, where \( |N| \) is the number of nodes. The number of actions that each node
4.4. Results

Figure 4.12: Average number of NTS frames with \(flag = 1\) sent in the simulation stage.

The number of NTS frames decreases quickly after the 1000-th transmission. When the number of NTS frames is less than the number of nodes, we find that some nodes choose to remain silent after negotiation instead of exploring possible actions. This helps reduce training complexity.

This experiment now uses the topology shown in Figure 4.14. The six nodes are divided into two disconnected cells. The aim here is to study whether the proposed Q-learning scheduler enables more than two concurrent transmissions. Figure 4.15 shows that the average throughput of the proposed algorithm is approximately 37 Mbps, which is around 150% higher than the average throughput of experiments that use single-hop topologies. The reason is that nodes learned that three concurrent transmissions are possible in this topology. For example, node-1 and node 2 have a bi-directional transmission, while node-4 and node-5 have a half-duplex transmission. As for CSMA, only half-duplex transmissions are possible in each
4.4. Results

Figure 4.13: Training time.

cell. Therefore, the average throughput achieved by CSMA also increases about 100%, which is 15 Mbps. As for TDMA$^1$ and TDMA$^2$, the average throughput does not change because nodes transmit one after another.

Figure 4.14: Topology with six nodes located in two cells.

4.4.4 Varying Channel Condition

the topology is the same as in Section 4.4.1, $\gamma = 0.75$ and learning rate $\alpha^1$. To vary the channel, the $\sigma$ is set to increase from 1 dB to 5 dB. Referring to Figure 4.16, as expected, the average throughput both decreases with $\sigma$. However, the average throughput of the proposed algorithm decreases slower than that of CSMA after $\sigma$ becomes 3 dB. The rate of decrease reduces from about 7 Mbps per dB to 1 Mbps per dB. On the contrary, for CSMA, the average throughput reduces at a constant rate of 1.7 Mbps per dB. The reason is that when $\sigma$ is small, e.g., less than 3 dB,
4.4. Results

Figure 4.15: Average throughput achieved by algorithms in a two cells topology.

if nodes use the proposed algorithm, they will use the highest data rate since the channel condition is relatively stable. Consequently, nodes will have more chance to receive a high reward because transmissions can be carried out at the highest data rate without frequent failures. When $\sigma$ is large, nodes learn to use a low data rate to ensure reliability. Recall that nodes that use CSMA choose data rates according to the number of successful transmissions. Hence, when channel condition deteriorates, the probability of successful transmissions also decreases, meaning nodes will use a lower data rate, which results in a lower average throughput. When nodes use TDMA$^1$ and TDMA$^2$, the average throughput achieved by TDMA$^2$ does not have any obvious decrease until $\sigma$ reaches 3 dB. This is because nodes choose the lowest data when they use TDMA$^2$, which requires a low SINR threshold. On the contrary, the average throughput achieved by TDMA$^1$ quickly decreases and becomes the lowest after $\sigma$ reaches 3 dB. This is because nodes choose the highest data rate even though the channel condition is highly unstable, which leads to transmission failures.
4.5 Conclusion

This chapter has presented a learning approach that allows nodes to jointly schedule links and set an appropriate data rate. The proposed approach is distributed and only requires information from neighboring nodes. The results show that nodes can achieve a 200% increase in average throughput after training as compared to when nodes use CSMA, and up to 300% increase in average throughput as compared to TDMA. Moreover, the proposed algorithm remains superior when channel gains vary between 30% to 100% of their original value.

A key technology that is widely used in wireless network is MIMO. In fact, IBFD can also be achieved using MIMO where some antenna elements can be used to cancel self-interference as well as interference from neighboring cells. Another consideration not considered in Chapter 3 is random traffic arrivals. To this end, the next chapter outlines a Q-learning scheduler that allocates antenna elements at nodes to remove interference as well as data transmissions or/and receptions.
Chapter 5

A DoF-Based Q-Learning Algorithm for WLANs

The DoF model has been widely used for scheduling the antenna elements of nodes [54, 78–81]. This is because it offers a significantly simpler representation as compared to the traditional matrix-based representation of a MIMO system [32]. Hence, it enables easier computation of sufficient condition that governs the number of active data streams that can be supported by a node. Also, as IBFD can also be realized using MIMO, see [58], it is thus interesting to consider using the DoF model for scheduling IBFD links.

Henceforth, this chapter outlines such a novel research direction. The system under consideration is a single cell WLAN with multiple clients that are associated to an AP. Both the AP and clients are MIMO capable and they are equipped with multiple antennas. They are also equipped with an IBFD radio. Data packets arrive randomly at the AP and clients, which are stored in queues with a limited size. In addition, APs and clients experience a random number of interfering streams from neighboring cells. Given this setup, this chapter aims to address the following problem: find the most suitable client(s) to poll in every time slot that maximizes
the number of data streams and also minimizes packet drops.

To illustrate the problem, consider the example shown in Figure 5.1 where one AP is associated with two clients $a$ and $b$. Both the AP and clients are MIMO and IBFD capable. Assume all three nodes are equipped with five antennas and are only able to store two packets. The AP has one data packet for each client. Client-$a$ has two data packets for the AP and client-$b$ has one data packet for the AP. Client-$a$ has three interfering streams from a neighboring cell and client-$b$ has four interfering streams. Using the DoF model [32], the AP can take one of the following actions: i) the AP downloads one packet to client-$b$, ii) the AP requests one client to upload, iii) the AP downloads to one client and requests another client to upload at the same time, or iv) the AP downloads to one client and the client uploads to the AP at the same time. The problem becomes easier if the AP knows the number of packets and interfering streams at clients. In this case, the AP can request client-$a$ to upload while downloading to client-$b$, which is the best choice because there will be three data streams. For any other actions, there will be fewer number of data streams. Moreover, client-$a$ may start dropping packets if it is not requested to upload its packets.

Figure 5.1: A WLAN with MIMO and full duplex capability. MIMO full duplex link is shown by double head blue arrows. Each node is experiencing interfering streams from neighboring cells, shown by orange arrows.

Collecting queue and interfering streams information from clients is time-consuming. Henceforth, this thesis will consider a learning approach whereby the AP only polls
up to two clients per time slot and updates the information of these clients. Then, the AP decides on an action that aims to maximize the number of data streams and also minimize packet drops due to buffer overflow. Henceforth, this chapter makes the following contributions:

1. It presents a Q-learning [113] based scheduling algorithm for centralized wireless networks, where all nodes are MIMO and IBFD capable. The proposed algorithm is able to schedule half/full-duplex MIMO links under the DoF model [32]. The proposed algorithm considers random interfering streams from nearby cells as well as random traffic loads.

2. The simulation results show that the proposed Q-learning based scheduling algorithm is able to achieve up to about 160% more average number of data streams as compared to a polling-based method which requires perfect knowledge of random interfering streams and queue state. The proposed algorithm has 15% less packets drops. When the system is saturated, where nodes always have packets to transmit, the proposed algorithm has similar performance as the polling method that randomly polls a client in each iteration.

This chapter is organized as follows. Section 5.1 presents the network model, DoF model, traffic model, interference model and problem definition. Section 5.2 presents a MDP model of the problem, and Section 5.3 presents the proposed Q-learning based algorithm. Section 5.4 outlines the evaluation methodology and also outline three polling-based methods used for benchmarking purposes. Lastly, simulation results are shown in Section 5.5. This chapter concludes in Section 5.6.

5.1 Preliminaries

5.1.1 Network Model

Time slot is indexed by $t$ and has length $\tau$. There is an AP, which has ID zero, that serve $|\mathcal{N}|$ clients from the set $\mathcal{N} = \{1, 2, \cdots, |\mathcal{N}|\}$. This chapter will use nodes
5.1. Preliminaries

to refer to both the AP and clients. All nodes have $K$ antennas, which they use for spatial multiplexing, and to cancel interference either from itself in the case of IBFD or from neighboring cells. Denote the number of antennas allocated for sending and receiving stream(s) at node $n$ as $A_n^S$ and $A_n^R$, respectively. The number of antennas used for self-interference cancellation at node $n$ is $A_n^\pm$, and the number of antennas used to cancel interfering streams from neighboring cells at a sending node is denoted as $A_n^{S-}$. A directional link from sender $i$ to receiver $j$ is denoted as $l(i,j)$.

5.1.2 DoF Model

Nodes use the DoF model [32] to allocate their antenna elements for transmission/reception of streams, and to cancel interfering streams. Briefly, the DoF of a node is equal to the number of antennas it has; hence, the DoF of nodes is equal to $K$. The DoFs or antennas of nodes are allocated as follows: (i) if a node is the transmitter of $x$ data streams, it consumes $x$ DoFs to cancel interference caused to unintended receive nodes, (ii) for a receiver, it consumes $x$ DoFs to cancel $x$ interfering streams, (iii) if a node is using IBFD to transmit $x$ streams, in order to receive any incoming streams, then it needs to consume an additional $x$ DoFs to cancel the interference caused by its outgoing $x$ streams, (iv) both the sender $i$ and receiver $j$ of link $l(i,j)$ consume $x$ DoFs to transmit and receive $x$ streams.

Formally, when the AP has a half-duplex link to client $n$, it must satisfy the following constraints:

$$A_0^S + A_0^{S-} \leq K,$$  \hspace{1cm} (5.1)

$$A_n^R + A_n^{S-} \leq K,$$  \hspace{1cm} (5.2)

where inequality (5.1) means the total number of antennas allocated at an AP for transmission plus the number of antennas used to cancel interfering streams from neighboring cells must not exceed $K$; we have a similar constraint for client $n$.

For a bi-directional full-duplex transmission, the AP and client $n$ must ensure
that,

\[ A^S_0 + A^R_0 + A^±_0 + A^S_-  \leq K, \quad (5.3) \]

\[ A^S_n + A^R_n + A^±_n + A^S_-  \leq K, \quad (5.4) \]

The main consideration in the above constraints is that a node has to cancel interference caused by themselves and from neighboring cells.

When the AP, client \( n \) and \( m \) have a full duplex relay transmission consisting of link \( l(n,0) \) and \( l(0,m) \), the following constraints must be satisfied,

\[ A^S_0 + A^R_0 + A^±_0 + A^S_-  \leq K, \quad (5.5) \]

\[ A^S_n + A^S_- + A^±_n \leq K, \quad (5.6) \]

\[ A^R_m + A^S_- \leq K, \quad (5.7) \]

where inequality (5.6) and (5.7) ensure the total number of antennas allocated for transmission/reception plus those for interference cancellation at the client \( m \) or \( n \) is no more than \( K \). In addition, client \( m \) has to also allocate antennas to cancel interference caused to client \( n \).

### 5.1.3 Traffic Model

Each client \( n \) has an upload and download queues with length \( Q^+_n \) and \( Q^-_n \), respectively. Note that a client’s upload queue is located at the client whilst its download queue is managed by the AP. At each time slot, the number of arrivals into a client’s upload and download queue follows the Poisson process,

\[ \Lambda(x) = e^{-\lambda \tau} (\lambda \tau)^x \frac{1}{x!} \quad (5.8) \]

where \( x \) is the number of streams that arrives in one time slot, and \( \lambda \) is the data stream arrival rate.
5.1.4 Interference

The AP and its associated clients observe random number of interfering streams originating from neighboring cells in each time slot $t$. The number of interfering streams at a node $n$ is denoted as $I^t_n$, and is represented by a Markov chain with a finite state space, denoted as $\mathcal{I}_n \in \{0, 1, 2, 3, \ldots, K\}$. The transition probability between state $I^t_n$ to $I^{t+1}_n$ is denoted as $P_n(I^{t+1}_n | I^t_n)$. Specifically, $P_n(I^t_n \leq j | I^{t-1}_n)$ represents the probability that the number of interfering streams at node-$n$ changes from $I^t_n - 1$ in the previous time slot to less than $j$ in the current time slot. In practice, the transition probability between states is obtained via measurement by each node; e.g., using IEEE 802.11k. It can then be computed using standard methods; see [136].

Table 5.1: Symbols and Description.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$l(i,j)$</td>
<td>A directed link between transmitter $i$ and receiver $j$.</td>
</tr>
<tr>
<td>$A^S_n$</td>
<td>The number of antennas allocated for transmission.</td>
</tr>
<tr>
<td>$A^R_n$</td>
<td>The number of antennas allocated for reception.</td>
</tr>
<tr>
<td>$A^\pm_n$</td>
<td>The number of antennas allocated to cancel self-interference.</td>
</tr>
<tr>
<td>$A^S_{-n}$</td>
<td>The number of antennas allocated to cancel interfering streams.</td>
</tr>
<tr>
<td>$Q^+_n$</td>
<td>Upload queue length of client $n$.</td>
</tr>
<tr>
<td>$Q^-_n$</td>
<td>Download queue length of client $n$.</td>
</tr>
<tr>
<td>$I^t_n$</td>
<td>The number of interfering streams.</td>
</tr>
</tbody>
</table>

5.1.5 Problem Definition

For a given time $t$, define the random vector $\mathbf{q}^t = \{(Q^+_n, Q^-_n) | \forall n \in \mathcal{N}\}$, and $\mathbf{I}^t = \{I^t_n | \forall n \in \mathcal{N}\}$. Let $\mathbf{u}^t$ be a vector of size $|\mathcal{N}|$ that indicates one or more devices that are polled by the AP in time slot $t$. The goal is to maximize the following long term reward:

$$ R = \lim_{T \to \infty} \frac{1}{T} \mathbb{E}_{\mathbf{q}^t, \mathbf{u}^t} \left[ \sum_{t=0}^{T-1} R^t(\mathbf{u}^t) \right] $$

(5.9)
5.2. An MDP Model

Here, \( R_t(u^t) \) is the reward obtained in time \( t \) given action \( u^t \); this reward will be defined precisely in Section 5.2. The expectation is taken with respect to the joint probability distribution between traffic arrival at each client and the number of interfering streams that are experienced by each client.

5.2 An MDP Model

An MDP can be described as a 4-tuple \( \{S, A, P(s, a), R(s, a)\} \), where the set \( S \) contains finite number of states. The set \( A \) is a finite set of actions. At each iteration or time step, an agent visits a state \( s \in S \), chooses and executes an action \( a \in A \). Then, the state \( s \) transitions to another state \( s' \) with probability \( P(s, a, s') \). The agent receives a reward \( R(s, a) \). The agent’s objective is to find the optimal policy \( \pi \) that maps each state \( s \) with a action \( a \) that maximizes the expected reward.

The said state, action, reward, and transition possibility are now defined formally:

1. **State.** Each state is represented by the following \( 2 \times |N| \) matrix:

\[
\begin{bmatrix}
q_1^+ & \cdots & q_n^+ \\
q_1^- & \cdots & q_n^- \\
\end{bmatrix}
\]  

(5.10)

where the binary variable \( q_n^+ \) and \( q_n^- \) represent the occupancy of the upload queue and download queue of client \( n \). That is, if \( Q_n^+ \) or \( Q_n^- \) is not zero, then \( q_n^+ \) or \( q_n^- \) is equal to one. Otherwise, we have \( q_n^+ = 0 \) or \( q_n^- = 0 \). The AP only updates \( q_n^+ \) or \( q_n^- \) in iteration \( t + 1 \) if client \( n \) is polled in iteration \( t \).

2. **Action.** The set of all available actions under state \( s \) is denoted as \( A(s) \). At each time slot, the AP has two types of actions: \( a^H \) and \( a^F \). When the AP takes action \( a^H = \{x, \theta\} \), it polls client \( x \) for upload (\( \theta = 1 \)) or download (\( \theta = -1 \)). On the other hand, action \( a^F = \{x, y\} \) means the AP polls client-\( x \) for upload, and client-\( y \) for download. If \( x = y \), then that means the AP and
client will use bi-directional full duplex transmission. Otherwise, the AP and the two clients have a relay full duplex transmission, which is from client \(x\) to the AP, and from the AP to client \(y\).

3. **Reward** The reward for action \(a^H\) or \(a^F\) is denoted as \(R(s, a^H)\) and \(R(s, a^F)\), respectively. They are calculated as,

\[
R(s, a^H) = \sigma q^+_x A^S_x - (1 - \sigma) \frac{\rho_x K}{T(x)} \tag{5.11}
\]

\[
R(s, a^F) = \sigma (q^+_x A^S_x + q^+_y A^S_y) - (1 - \sigma) K \left[ \frac{\rho_x}{T(x)} + \frac{\rho_y}{T(y)} \right] \tag{5.12}
\]

where \(\rho_x\) and \(\rho_y\) denote the number of packets that are discarded from client-\(x\) and \(y\) since they are last polled by the AP. The variable \(T(x)\) represents how many slots since client \(x\) has been polled by the AP. The weight \(\sigma\) has range [0, 1]. Both \(R(s, a^H)\) and \(R(s, a^F)\) increase with the total number of transmitted streams, and decreases with the total number of dropped packets. When weight \(\sigma\) is near one, the AP will prefer to maximize the number of data streams. Alternatively, when the weight \(\sigma\) is near zero, the AP aims to prevent packet drops.

4. **State transition possibility**. The state transition possibility is denoted as \(P(s, a^H, s')\) and \(P(s, a^F, s')\) which are calculated as,
5.2. An MDP Model

\[ P(s, a^H, s') = \left\{ \begin{align*}
& P_x(I_x^t \leq (K - Q^+_x)|I_x^{t-1}) \\
& \times P_0(I_0^t \leq (K - Q^+_x)|I_0^{t-1})\Lambda(0) + \Lambda(0), \text{ if } q^+_x \in s' = 0. \\
& 2 - P_x(I_x^t \leq (K - Q^+_x)|I_x^{t-1}) \\
& \times P_0(I_0^t \leq (K - Q^+_x)|I_0^{t-1})\Lambda(0) - \Lambda(0), \text{ if } q^+_x \in s' = 1. \\
& P_x(I_x^t \leq (K - Q^-_x)|I_x^{t-1}) \\
& \times P_0(I_0^t \leq (K - Q^-_x)|I_0^{t-1})\Lambda(0) + \Lambda(0), \text{ if } q^-_x \in s' = 0. \\
& 2 - P_x(I_x^t \leq (K - Q^-_x)|I_x^{t-1}) \\
& \times P_0(I_0^t \leq (K - Q^-_x)|I_0^{t-1})\Lambda(0) - \Lambda(0), \text{ if } q^-_x \in s' = 1. 
\end{align*} \right. \]

When the AP chooses to poll client \( x \) for upload or download, if the upload or download queue of client \( x \) is empty, the state transitions only when there is packet arrival. Otherwise, the state will only transition if the upload queue or download queue of the client has been emptied after communicating with the AP, and no new packet arrives in the current time slot. Formally,

\[ P(s, a^F, s') = \left\{ \begin{align*}
& P_0(I_0^t \leq (K - 2Q^+_x - 1)|I_0^{t-1}) \\
& \times P_x(I_x^t \leq (K - Q^+_x)|I_x^{t-1})\Lambda(0) + \Lambda(0), \text{ if } q^+_x \in s' = 0 \\
& 2 - P_0(I_0^t \leq (K - 2Q^+_x - 1)|I_0^{t-1}) \\
& \times P_x(I_x^t \leq (K - Q^+_x)|I_x^{t-1})\Lambda(0) - \Lambda(0), \text{ if } q^+_x \in s' = 1. \\
& P_0(I_0^t \leq (K - Q^-_y - 2)|I_0^{t-1}) \\
& \times P_y(I_y^t \leq (K - Q^-_y)|I_y^{t-1})\Lambda(0) + \Lambda(0), \text{ if } q^-_y \in s' = 0 \\
& 2 - P_0(I_0^t \leq (K - Q^-_y - 2)|I_0^{t-1}) \\
& \times P_y(I_y^t \leq (K - Q^-_y)|I_y^{t-1})\Lambda(0) - \Lambda(0), \text{ if } q^-_y \in s' = 1.
\end{align*} \right. \]

When the AP chooses to poll client \( x \) for upload, and client \( y \) for download, if the upload queue of client \( x \) and the download queue of client \( y \) are empty, the
AP and the polled clients will have zero data streams. The state transitions when new packets arrive at the queue of polled clients in the current time slot. Otherwise, the state transitions if,

(a) the upload queue of client-\(x\) has been emptied after communicating with the AP, and no new packets arrive at the upload queue of client-\(x\) in the current time slot.

(b) the download queue of client-\(y\) has been emptied after communicating with the AP, and no new packets arrive at the download queue of client-\(y\) in the current time slot.

Let \(\pi\) denote the policy that maps a state \(s\) with an action \(a\). Let \(V_\pi(s)\) be the value function that returns the expected reward at state \(s\). In particular, the value of \(V_\pi(s)\) is calculated as follows,

\[
V_\pi(s) = \mathbb{E}\{R(s, \pi(s)) + \gamma R(s', \pi(s')) + \gamma^2 R(s'', \pi(s'')) \cdots | s\}.
\] (5.14)

where the \(\gamma\) denotes the discount factor, \(0 < \gamma < 1\). To compute the optimal value of \(V_\pi\), denoted as \(V^*\), Equ. (5.14) can be solved via the Bellman equation as,

\[
V^*(s) = \max_{a \in \mathcal{A}(s)} \left[ R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s, a, s') V^*(s') \right].
\] (5.15)

The optimal policy \(\pi^*\) for all states \(s \in \mathcal{S}\) is given by,

\[
\pi^*(s) = \arg \max_{a \in \mathcal{A}(s)} \left[ R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s, a, s') V^*(s') \right].
\] (5.16)

The optimal policy \(\pi^*\) requires \(|\mathcal{A}(s)||\mathcal{S}|^2\) iterations to be found via a model-based approach such as the Value Iteration method \([112]\), which requires the probability of state transition. The next section proposes a model-free approach.
5.3 A Q-Learning Based Link Scheduler

Agents using Q-Learning [113] maintain a so called \( Q \)-factor, denoted as \( Q(s, a) \), for each state \( s \) and action \( a \). The value of \( Q(s, a) \) is calculated using,

\[
Q(s, a) = (1 - \alpha)Q(s, a) + \alpha \left[ R(s, a) + \gamma \max_{a' \in A(s)} Q(s', a') \right]
\] (5.17)

with learning rate \( \alpha \), which controls how much the agent weighs previous and current information. When each state \( s \in S \) is visited infinitely by an agent, the value of \( Q(s, a) \) converges to the optimal value [113]. Hence, after training, an agent is able to obtain the optimal action when it is in state \( s \) by computing \( \arg \max_{a \in A(s)} Q(s, a) \).

Three algorithms are used to allocate the antennas of nodes; these algorithms use the constraints in Section 5.1.2. Algorithm 10 is used to allocate antennas when the AP has selected action \( a^H \). Firstly, the AP and the selected client assign antennas to null interfering streams; i.e., the number of assigned antennas is equal to the number of interfering streams. Then, if a client is requested to upload, the client allocates the corresponding number of transmit antennas as shown in line 4 of Algorithm 10. The AP allocates an equal number of antennas for receiving as the number of transmitting antennas allocated by the client. If the client is requested to download, the client allocates a number of receiving antennas as shown in line 6 of Algorithm 10. The AP allocates an equal number of antennas for transmissions as the number of antennas dedicated to reception by the client.

\begin{algorithm}
\caption{Antennas allocation for half-duplex transmissions.}
\begin{algorithmic}[1]
\State \( A_{x}^{-} = I_{x} \);  
\State \( A_{0}^{-} = I_{0} \);  
\If{\( \theta = 1 \)}  
\State \( A_{x}^{+} = A_{0}^{-} = \min \{ Q_{x}^{+}, K - A_{x}^{-}, K - A_{0}^{-} \} \);  
\EndIf  
\If{\( \theta = -1 \)}  
\State \( A_{x}^{-} = A_{0}^{+} = \min \{ Q_{0}^{-}, K - A_{x}^{+}, K - A_{0}^{+} \} \);  
\EndIf
\end{algorithmic}
\end{algorithm}

Algorithm 11 is for antenna allocation when the AP has selected to execute a bi-
directional transmission with client-x. Firstly, the AP and client-x allocate antennas to cancel interfering streams. Then, as shown in line 3 of Algorithm 11, client-x allocates transmitting antennas and the AP allocates receiving antennas. Based on the constraints shown in Section 5.1.2, the maximum number of antennas that client-x can use for uploading while reserving at least one antenna for downloading is equal to \( \frac{K - A_S^x}{2} \). The maximum number of antennas that the AP can use for uploading while reserving at least one antenna for downloading is equal to \( K - 2 - A_R^0 \). Lastly, as shown in line 4 of Algorithm 11, client-x allocates receiving antennas and the AP allocates transmitting antennas. Based on the constraints shown in Section 5.1.2, the maximum number of antennas that the client-x can use for downloading is equal to \( K - A_S^x - A_R^0 \). The maximum number of antennas that the AP can use for downloading is equal to \( \frac{K - A_S^y - A_R^y}{2} \).

Algorithm 11: Allocate antennas for bi-directional transmission.

1. \( A_S^x = I^x \);
2. \( A_S^0 = I^0 \);
3. \( A^\pm_x = A_S^x = A_R^0 = \min \left\{ q_x^+, \frac{K - 1 - A_S^x}{2}, (K - 2 - A_S^y) \right\} \);
4. \( A^\pm_0 = A_S^0 = A_R^x = \min \left\{ q_x^-, (K - A_S^x - A_S^y - A_R^x) \right\} \);

Algorithm 12 is used to allocate antennas when the AP has selected to execute a relay transmission between client-x and y. Firstly, the AP and the two selected clients allocate antennas to cancel interfering streams. Then, as shown in line 4 of Algorithm 12, client-x allocates antennas for transmission and the AP allocates antennas for reception. Based on the constraints shown in Section 5.1.2, the maximum number of antennas that client-x uses for uploading is equal to \( \frac{K - A_S^x}{2} \). The maximum number of antennas that the AP uses for uploading while reserving at least one antenna for downloading is equal to \( K - 2 - A_R^0 \). Lastly, as shown in line 5 of Algorithm 12, client-y allocates antennas for receiving and the AP allocates antennas for transmitting. Based on the constraints shown in Section 5.1.2, the maximum number of antennas that client-y uses for downloading is equal to \( K_y - A_S^y \). The maximum number of antennas that the AP uses for downloading is
equal to \( \frac{K - A_0^S - A_R^S}{2} \).

**Algorithm 12:** Antennas allocation for relay transmission.

1. \( A_x^{S-} = I_x^t \);
2. \( A_y^{S-} = I_y^t \);
3. \( A_0^{S-} = I_0^t \);
4. \( A_x^\pm = A_x^S = A_0^R = \min \left\{ q_x^+, \frac{K - A_x^{S-}}{2} - (K - 2 - A_0^{S-}) \right\} \);
5. \( A_0^\pm = A_0^S = A_y^R = \min \left\{ q_y^-, (K_y - A_y^{S-}) - \frac{K - A_0^{S-} - A_R^R}{2} \right\} \);

### 5.4 Evaluation

The simulator used for experiments is implemented in C#. The AP is located at the center of a \( 100 \times 100m^2 \) area, and clients are randomly placed around the AP. Table 5.2 shows the simulation parameters. The AP is trained with a fixed learning rate and discount factor. The AP chooses an action using \( \epsilon \)-greedy. Specifically, the AP has a probability \( \epsilon \) to choose an available action under the current state. Otherwise, with probability \( (1 - \epsilon) \), the AP chooses the greedy action, which has the highest Q-factor. During training, the packets arrival rate \( \lambda \) is set to \( K_2 \times N \). This gives a traffic load that is within the capacity of the tested network. During simulation, the packets arrival rate \( \lambda \) is set to two, except for the experiments in Section 5.5.5.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area size</td>
<td>100m × 100m</td>
</tr>
<tr>
<td>Transmit power ( P_t )</td>
<td>100 mW</td>
</tr>
<tr>
<td>Antenna gain ( G_r ) and ( G_t )</td>
<td>2 dBi</td>
</tr>
<tr>
<td>Number of antennas ( K )</td>
<td>15</td>
</tr>
<tr>
<td>Discount factor ( \gamma )</td>
<td>0.99</td>
</tr>
<tr>
<td>Learning rate ( \alpha )</td>
<td>0.1, 0.2, 0.3... , 0.9</td>
</tr>
<tr>
<td>Weight ( \sigma )</td>
<td>0.1, 0.2, 0.3... , 0.9</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>0.2</td>
</tr>
<tr>
<td>Simulation iterations</td>
<td>10000</td>
</tr>
</tbody>
</table>

The proposed centralized Q-learning based scheduler is compared against three
polling-based MACs. In the first polling-based MAC, denoted as \textit{Polling-Per}, the AP collects the current queue size and the number of interfering streams from all clients at the beginning of each iteration. Then, based on the collected information, the AP calculates how many data streams each client has and proceeds to poll the client that has the most data streams to upload or download. Hence, the performance of \textit{Polling-Per} is theoretical upper bound for a half-duplex MAC. In the second polling-based MAC, denoted as \textit{Polling-Ran}, the AP randomly selects a client for upload or download in each iteration. In the last Polling-based MAC denoted as \textit{Polling-Seq}, the AP polls one client in each iteration sequentially based on the client’s ID. In all simulations, the number of iterations is set to 10000.

The simulator records the following metrics:

1. \textit{Average data streams}. This is equal to the total number of data streams divided by the total number of simulation iterations.

2. \textit{Average packets drops}. We record the average total number of discarded packets divided by the total number of simulation iterations.

\section*{5.5 Results}

The first experiment studies the optimal value of parameters that impacts the training. Hence, Section 5.5.1 studies the impact of learning rate $\alpha$. Then, Section 5.5.2 investigates different weight $\sigma$ in Equ. 5.11 and Equ. 5.12. Section 5.5.3 studies the Q-learning algorithm under various topologies. Section 5.5.4 studies the impact of DoF and maximum queue size. Lastly, Section 5.5.5 investigates various packets arrival rates and their impact on performance.

\subsection*{5.5.1 Learning Rate}

The AP is associated with five randomly placed clients. The AP is trained with a learning rate $\alpha$ that starts from 0.1 to 0.9, and changed with an interval of 0.1. The
maximum queue size is limited to five. The value of $\lambda$ is set to two. From Figure 5.2, we observe that the learning rate $\alpha$ has no obvious impact. The average number of data streams remains at around 5.6 and the average number of packet drops is about 10. The reason is that when the learning rate $\alpha$ is high, the AP updates Q-factors based on the immediate reward. On the contrary, when the learning rate $\alpha$ is low, the AP updates Q-factors based on the received reward in the previous iteration. However, as shown by the state transition probability in Section 5.2, the current state is not solely determined by the action that the AP has taken in the previous iteration. The AP may stay in the same state for multiple iterations. When $\epsilon$ is set to 0.2, the AP intends to choose the greedy action. Hence, the immediate reward and the reward from the last iteration may represent the same state and action. Therefore, the AP learns the same policy regardless of the value of the learning rate $\alpha$.

Figure 5.2: Average data streams and packet drops after training learning Rate $\alpha$.

### 5.5.2 Reward Function Weight

In this experiment, the AP is also associated with five randomly placed clients. The maximum queue size is limited to five. The learning factor $\alpha$ is set to 0.5. The weight $\sigma$ starts from 0.1 to 0.9 with an interval of 0.1, and $\lambda = 2$. From Figure 5.3,
the weight $\sigma$ does not influence the average throughput or the average the number of packet drops. When $\sigma$ is high, the AP receives a high reward if there are many data streams. With more data streams at each iteration, fewer packets will be dropped. When $\sigma$ is low, the AP receives a high reward if the number of packets drops is low. However, there is no conflict between preventing packets drops and establishing more data streams. With fewer packets drops at each iteration, the AP must have enabled more data streams. Consequently, adjusting the weight $\sigma$ does not influence the policy learned by the AP.

![Figure 5.3: Average data streams and packet drops after training with different weight $\sigma$.](image.png)

### 5.5.3 Number of Clients

In this experiment, we increase the number of clients from two to six. We set the learning rate $\alpha$ and weight $\sigma$ to 0.5. The maximum queue size is five. The $\lambda$ is set to two. From Figure 5.4 and 5.5, we see the the Q-Learning algorithm has higher average number of data streams and lower packets drops than other methods. The average number of data streams achieved by the Q-Learning algorithm is between 5.6 to 6. The average number of data streams achieved by the three polling methods is 4.7. The reason is that when all clients have a large number of interfering streams, the number of data streams is low in polling-based methods because the AP is only
able to execute half-duplex transmissions with one client. Hence, the AP may have
spare antennas that are either not used to transmit/receive or to cancel interfering
streams. As for the Q-Learning algorithm, if a client has a large number of interfering
streams, the AP learns to also poll another client to execute a relay transmission.
Hence, the number of data streams is increased. Consequently, the Q-Learning
algorithm achieves better average data streams. The average packets drop increases
with the number of clients in all methods. The reason is that the overall traffic load
increases with the number of clients but the network capacity remains the same. The
average packet drops achieved by the Q-Learning algorithm is about one packet per
slot lower than polling methods because it always has a higher average number of
data streams under all topologies.

![Image](Fig5.4.png)

Figure 5.4: Average data streams in topologies with varying number of clients.

### 5.5.4 Maximum Queue Size

In this experiment, five clients are randomly placed around the AP. The maximum
queue size ranges from 5 to 30 with an interval of five. The learning rate $\alpha$ and weight
$\sigma$ are both set to 0.5, and $\lambda$ is set to two. From Figure 5.6 and 5.7, the performance
of all methods increases with the maximum queue size before the maximum queue
size reaches 20. The reason is that both the AP and clients have fifteen antennas.
Figure 5.5: Average packet drops in topologies with varying number of clients.

Each data stream only carries one packet. Hence, the number of data streams is limited by how many packets are stored in the queue. We also observe that when the maximum queue size is 5 or 10, the Q-Learning algorithm has a better performance because it allows the AP to execute full-duplex transmissions, which draws packets from two queues. However, after the maximum queue size changes to 20 or higher, the number of data streams in each iteration is limited by the available antennas. Hence, Polling-Per has an average number of data streams around 14 per iteration. The other two polling based methods have an average number of data streams of around 12 per iteration. On the other hand, the average number of data streams achieved by Q-learning algorithm decreases to 12 per iteration after the maximum queue size is limited to 20. The reason is that when nodes have a large queue, the state as observed by the AP remains the same. Recall that the state represents the occupancy of clients’ queue. Hence, when there are many packets in the queue of clients, the state remains the same for a large number of iterations. For example, assume a client has a buffer with 30 packets and that these packets can be cleared in two iterations if they have no interfering streams. Hence, the state will reflect that the client’s queue as occupied for two iterations. However, with interfering streams, these packets will take a longer time to transmit. Moreover, in both scenarios, new packets may arrive. Consequently, when the AP finds that all clients’ queue has at
least once packet, it will randomly choose an action. Thus, the performance of the Q-Learning algorithm becomes close to Polling-Ran. As shown in Figure 5.6 and 5.7, Q-Learning algorithm, Polling-Ran and Polling-Seq have similar performance; both in terms of the average number of data streams and packet drops.

![Figure 5.6: Average data streams under various maximum queue sizes.](image)

![Figure 5.7: Average packet drops under various maximum queue sizes.](image)

5.5.5 Packets Arrival Rate

In this experiment, the network has one AP and five randomly placed clients. The learning rate $\alpha$ and weight $\sigma$ are both set to 0.5. The value of $\lambda$ is set to $0.25 \times, 0.5 \times,$
5.6 Conclusion

This chapter has presented a Q-learning based algorithm that is able to schedule half/full duplex links under DoF model. The proposed learning algorithm is able to
Figure 5.8: Average data streams under different packets arrival rates.

Figure 5.9: Average packet drops under different packets arrival rates.
allocate antenna resources to eliminate the influence of random interfering streams from nearby cells. It is also applicable to random traffic loads but does not require queue state information, which helps reduce signaling overheads.
Conclusion

This thesis has investigated multiple link scheduling approaches for various wireless networks. Its key aim is to schedule links in wireless networks where nodes are capable of transmitting and receiving simultaneously within the same frequency range, aka IBFD. This thesis has also considered link scheduling in wireless networks where nodes have MIMO capability. Both IBFD and MIMO have the most potential to increase the capacity of current wireless networks, especially if they are coupled with a link scheduler. In particular, a link scheduler controls the set of active links at any given time and hence, it has a direct impact on network capacity.

To date, there are many link scheduling works. Works related to IBFD only focus on enabling bi-directional and relay transmission modes, or reducing the interference between primary and secondary transmissions. Also, link scheduling works that consider MIMO-capable nodes do not support IBFD. Apart from that, existing link scheduling works aim to maximize the number of links in each time slot, and assign active links with a common duration and data rate. In addition, most existing link scheduling works assume fixed channel condition, or perfect channel state information.

With respect to the research questions outlined in Chapter 1, this thesis makes the following conclusions:
1. ‘Affectedness’ is an effective metric that can be used to construct a schedule. Chapter 3 presents three heuristic algorithms that use ‘affectedness’ to construct a schedule. These algorithms are capable of assigning simultaneously active links with a different activation duration and data rate. In addition, the second and third algorithms are able to determine whether full-duplex transmissions or multiple activated links are beneficial in terms of reducing completion time in a given time slot. As shown by the results in Chapter 3.5, the overall completion time can be reduced by 68% as compared to scheduling links individually. Moreover, our algorithms reduce the completion time by 13% as compared to existing scheduling methods.

2. It is beneficial to incorporate machine learning techniques into link scheduling algorithms. These techniques can help determine the optimal data rate for transmissions under varying channel condition. Chapter 4 outlines a distributed Q-learning based link scheduling algorithm that enables nodes to form full-duplex transmissions in a distributed manner through a short negotiation procedure. In addition, nodes are able to determine the optimal data rate for each transmission without carrying out an estimation process, considering both interference from neighbours and varying channel gains. The result shows the average throughput is triple that of Carrier Sense Multiple Access (CSMA), and up to quadruple the average throughput of Time Division Multiple Access (TDMA). Moreover, the proposed link scheduler in Chapter 3 remains superior when channel gains vary significantly from their average value.

3. The DoF model of [32] can be used to allocate antennas of nodes to enable IBFD links. Chapter 5 outlines a centralized Q-learning based link scheduling algorithm that schedules both half and full duplex links under the DoF model. It uses three antenna resources allocation algorithms derived from constraints pertaining to the DoF model to allocate antennas for transmissions, receptions and/or interference cancellation. The simulation results show that about 60%
increases in average number of data streams can be achieved by this centralized Q-learning based link scheduling algorithm, compared to polling methods which requires perfect knowledge about the network.

4. Nodes are able to learn to reduce packet drops under random traffic arrival rates using machine learning techniques. Specifically, Chapter 5 outlines a centralized Q-learning based link scheduling algorithm that enables an AP to learn to minimize packet drops. It requires only knowledge of clients that have been polled thus far. The simulation results show that the proposed algorithm in Chapter 5 reduces packet drops by about 15%, compared to a polling based methods, which requires queue information from all clients. Lastly, when all clients have saturated queues, the performance of the proposed algorithm converges to that of the random polling method.

There are many future research directions. As shown in Chapter 4, the proposed distributed Q-learning based scheduling algorithm requires a negotiation procedure. However, the negotiation procedure requires all nodes to be located within each other’s transmission range. One possible future research direction is to extend this negotiation procedure to multi-hop networks. Another possible future research direction is to improve the training methods in Chapter 5. The proposed centralized Q-learning based scheduling algorithm has a low training efficiency in some scenarios. Specifically, the AP can only visit a small portion of all possible states. Thus, future research may develop new training methods that help the AP discover new states quickly. Lastly, this thesis has not taken advantage of relays where an AP uses a nearby client as a relay to help improve communication with a far away client.
Bibliography


[98] S. Oashi and M. Bandai, “Performance of medium access control protocols for full-duplex wireless LANs,” in *IEEE 9th Asia-Pacific Symposium on Informa-
tion and Telecommunication Technologies, (Santiago and Valparaiso, Chile), pp. 1–4, Nov 2012.


