2013

Minimum latency broadcast algorithms for wireless sensor networks

Dianbo Zhao

University of Wollongong

Recommended Citation
UNIVERSITY OF WOLLONGONG

COPYRIGHT WARNING

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

Copyright owners are entitled to take legal action against persons who infringe their copyright. A reproduction of material that is protected by copyright may be a copyright infringement. A court may impose penalties and award damages in relation to offences and infringements relating to copyright material. Higher penalties may apply, and higher damages may be awarded, for offences and infringements involving the conversion of material into digital or electronic form.
Minimum Latency Broadcast Algorithms for Wireless Sensor Networks

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

PhD

from

THE UNIVERSITY OF WOLLONGONG

by

Dianbo Zhao
Bachelor of Science (Applied Physics)

SCHOOL OF ELECTRICAL, COMPUTER
AND TELECOMMUNICATIONS ENGINEERING
2013
Abstract

Broadcast is a fundamental operation in Wireless Sensor Networks (WSNs). Given a source node with a packet to broadcast, the aim is to propagate the packet to all nodes in an interference-free manner whilst incurring minimum latency. This problem, called Minimum Latency Broadcast Scheduling (MLBS), has been studied extensively in wireless ad-hoc networks, whereby nodes remain on all the time, and has been shown to be NP-hard. However, only a few studies have addressed this problem in the context of duty-cycled WSNs. In these WSNs, nodes do not wake-up simultaneously, and hence, not all neighbors of a transmitting node will receive a broadcast message at the same time. Unfortunately, this Minimum Latency Broadcast Scheduling problem in Duty-Cycled WSNs (MLBSDC) remains NP-hard and multiple transmissions may be necessary due to different wake-up times. Moreover, existing studies addressed the MLBSDC problem only over the idealistic interference model, i.e., the RTS/CTS interference model, in which, if two or more nodes transmit simultaneously to a single node, collision occurs and thereby, corrupting the message. However, this idealistic interference model does not take into account the interference from transmissions outside a receiver’s transmission range.

This thesis, therefore, investigates the MLBSDC problem under different interference models, i.e., the RTS/CTS, protocol, and physical interference model. Different from the RTS/CTS interference model, the other two models reflect the fact that the successful reception of a message is subject to interference from transmissions outside the receiver’s transmission range. This thesis proposes a series of approximation algorithms. Specifically, the main contributions of this thesis are as follows.
This thesis contributes two approximation algorithms, called BS-1 and BS-2, for the MLBSDC problem under the RTS/CTS interference model. In particular, BS-2 produces the best constant approximation ratio of $13T$ in terms of broadcast latency, as compared to other proposed algorithms. Here, $T$ denotes the number of time slots in a scheduling period.

Apart from that, this thesis outlines one centralized greedy heuristic algorithm and its distributed implementation, called CEN and DIS respectively, for the MLBSDC problem under the RTS/CTS interference model. The centralised version, i.e., CEN, produces a ratio of $(\Delta - 1)T$ in terms of broadcast latency, where $\Delta$ is the maximum degree of nodes. Extensive experimental results show that the broadcast latency of CEN and DIS is near optimal. In particular, compared to OTAB, the best broadcast scheduling algorithm to date, the broadcast latency and number of transmissions achieved by CEN are about $\frac{1}{5}$ and $\frac{1}{2}$ that of OTAB, respectively.

This thesis also contains two constant approximation algorithms for the MLBSDC problem under the protocol interference model. In particular, this thesis contains the first studies on the MLBSDC problem under the protocol interference model. The proposed algorithms, called IABBS and IAEBS produce a $O(\rho^2)$-approximate solution with respect to broadcast latency, where $\rho$ is the ratio between the interference and transmission range.

Finally, this thesis outlines the first distributed algorithm, called HBA, for the MLBSDC problem under the physical interference model. Furthermore, HBA gives a constant ratio in terms of the broadcast latency and number of transmissions. The performance of HBA is evaluated under different network configurations and the results show that the latencies achieved by HBA are much lower than existing schemes. In particular, the broadcast latency achieved by HBA is only $\frac{1}{2}$ that of the tree-based algorithm.
Statement of Originality

This is to certify that the work described in this thesis is entirely my own, except where due reference is made in the text.

No work in this thesis has been submitted for a degree to any other university or institution.

Signed

Dianbo Zhao

27th May, 2013
Acknowledgments

I would like to take this opportunity to thank people who guided and supported me during my research. Without their contributions, this research would not have been possible.

First of all, I would like to thank my supervisor, A/Prof. Kwan-Wu Chin. I thank him for his advice and support throughout my PhD. He taught me a lot on how to do research and gave me so many valuable advices on my research topic. He was always patient and helpful whenever his assistance is needed, and he has contributed much of his valuable time to improving my academic writing skills. I also highly appreciate his faith in me, allowing me to select my favorite research directions in these years.

I would also like to thank my co-supervisor: Dr. Raad Raad. I greatly appreciate his support and encouragement on my work, and also his willingness to bend his schedules to accommodate meetings and seminars.

Lastly, I would like to thank my family members for all their love and support. I am grateful to my parents. Without their love, I would not have reached this milestone and this thesis would not have happened. I am forever indebted to my wife, Bingyu Wang. She has been incredibly supportive of me through the ups and downs over the past years. I cannot thank her enough for being such a wonderful wife.

This thesis is to all people who have helped and are helping me.
# Contents

1 Introduction 

1.1 Duty-Cycled Wireless Sensor Networks ................. 2  
1.2 Broadcast ................................................... 4  
1.3 Problem Space and Motivation ............................. 7  
1.4 Contributions ................................................ 8  
1.5 Publications .................................................. 11  
1.6 Thesis Structure ............................................. 12

2 Literature Review 

2.1 Broadcast over Always-on Wireless Networks ............. 13  
2.1.1 Probability Based Methods ................................. 14  
2.1.2 Area Based Methods ....................................... 14  
2.1.3 Neighbor Knowledge Methods ............................ 15  
2.2 MLBS over Always-on Wireless Networks .................. 18  
2.2.1 Tree Based Algorithms ................................... 18  
2.2.2 Greedy Heuristic Algorithms ............................. 26  
2.3 MLBS over Duty-cycled WSNs ............................... 29  
2.3.1 Unicast Based Methods .................................. 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2.3 Distributed (DIS) Broadcast Algorithm</td>
<td>75</td>
</tr>
<tr>
<td>4.3 Evaluation</td>
<td>82</td>
</tr>
<tr>
<td>4.3.1 Results</td>
<td>83</td>
</tr>
<tr>
<td>4.4 Conclusion</td>
<td>90</td>
</tr>
<tr>
<td>5 Approximation Algorithms under the Protocol Interference Model</td>
<td>91</td>
</tr>
<tr>
<td>5.1 Preliminaries</td>
<td>92</td>
</tr>
<tr>
<td>5.1.1 Network Model</td>
<td>92</td>
</tr>
<tr>
<td>5.1.2 Graph Definitions and Theories</td>
<td>93</td>
</tr>
<tr>
<td>5.1.3 Key Observation</td>
<td>94</td>
</tr>
<tr>
<td>5.2 Proposed Algorithms</td>
<td>95</td>
</tr>
<tr>
<td>5.2.1 IABBS</td>
<td>96</td>
</tr>
<tr>
<td>5.2.2 IAEBS</td>
<td>103</td>
</tr>
<tr>
<td>5.2.3 Analysis</td>
<td>106</td>
</tr>
<tr>
<td>5.3 Evaluation</td>
<td>109</td>
</tr>
<tr>
<td>5.3.1 Results</td>
<td>110</td>
</tr>
<tr>
<td>5.4 Remarks on Duty-Cycled WSNs</td>
<td>112</td>
</tr>
<tr>
<td>5.5 Conclusion</td>
<td>114</td>
</tr>
<tr>
<td>6 Distributed Algorithm under the Physical Interference Model</td>
<td>115</td>
</tr>
<tr>
<td>6.1 Preliminaries</td>
<td>116</td>
</tr>
<tr>
<td>6.1.1 Network Model</td>
<td>116</td>
</tr>
<tr>
<td>6.1.2 Definitions and Theories</td>
<td>118</td>
</tr>
<tr>
<td>6.2 Distributed Broadcast Schedule</td>
<td>121</td>
</tr>
<tr>
<td>6.2.1 Broadcast Structure</td>
<td>121</td>
</tr>
<tr>
<td>6.2.2 Broadcast Scheduling</td>
<td>125</td>
</tr>
</tbody>
</table>
List of Figures

1.1 A typical WSN architecture ................................................. 1
1.2 Power consumption of sub-systems on a sensor node [22] .......... 2
1.3 An example where node A, B and C adopt an asynchronous schedule 4
1.4 An example of broadcast ................................................... 5
1.5 Bounded interference model ............................................... 7

3.1 The topology of network $G$ ................................................. 43
3.2 An illustration of BS-1. The label (rec, tr) denotes the reception and transmission time of a node. .................................. 44
3.3 An example to illustrate BS-2. A square indicates a dominator. .. 49
3.4 Broadcast latency under different network sizes ...................... 56
3.5 Transmission ratio under different network sizes .................... 57
3.6 Broadcast latency under different transmission radii ............... 58
3.7 Transmission ratio under different transmission radii ............. 58
3.8 Broadcast latency under different duty cycles ...................... 59
3.9 Transmission ratio under different duty cycles ..................... 60

4.1 An illustration of OPT. The label (rec, tr) denotes the reception and transmission time of a node ............................... 68
<table>
<thead>
<tr>
<th>Number</th>
<th>Figure Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Graph $G_X$ with $X = {v_1, v_6, v_7}$ and $N_X = {v_2, v_3, v_5, v_8}$</td>
<td>70</td>
</tr>
<tr>
<td>4.3</td>
<td>Graph $G_Y$ with $Y = {v_1, v_7}$ and $N_X = {v_2, v_3, v_5, v_8}$</td>
<td>72</td>
</tr>
<tr>
<td>4.4</td>
<td>An illustration of CEN</td>
<td>72</td>
</tr>
<tr>
<td>4.5</td>
<td>An illustration of data format used in DIS</td>
<td>76</td>
</tr>
<tr>
<td>4.6</td>
<td>An illustration of DIS</td>
<td>81</td>
</tr>
<tr>
<td>4.7</td>
<td>Broadcast latency under different number of nodes</td>
<td>84</td>
</tr>
<tr>
<td>4.8</td>
<td>Number of transmissions under different number of nodes</td>
<td>85</td>
</tr>
<tr>
<td>4.9</td>
<td>Broadcast latency under different transmission radii</td>
<td>86</td>
</tr>
<tr>
<td>4.10</td>
<td>Number of transmissions under different transmission radii</td>
<td>86</td>
</tr>
<tr>
<td>4.11</td>
<td>Broadcast latency under different duty cycles</td>
<td>87</td>
</tr>
<tr>
<td>4.12</td>
<td>Number of transmissions under different duty cycles</td>
<td>88</td>
</tr>
<tr>
<td>4.13</td>
<td>Performance of CEN, DIS versus OTAB</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>An example of two simultaneous transmissions</td>
<td>95</td>
</tr>
<tr>
<td>5.2</td>
<td>An example wireless network with $r_T = 1$ and $r_I = 2$. There are 13 nodes randomly deployed in a $4 \times 5$ rectangle area. Solid lines denote transmission range. Dotted lines denote the interference range.</td>
<td>96</td>
</tr>
<tr>
<td>5.3</td>
<td>Broadcast tree $T_b$ of IABBS. The label $(rec, tr)$ denotes the reception and transmission time of a node, and a square indicates a dominator.</td>
<td>99</td>
</tr>
<tr>
<td>5.4</td>
<td>Broadcast tree $T_b$ of IAEBS.</td>
<td>105</td>
</tr>
<tr>
<td>5.5</td>
<td>Broadcast latency under different number of nodes</td>
<td>110</td>
</tr>
<tr>
<td>5.6</td>
<td>Transmission ratio under different number of nodes</td>
<td>111</td>
</tr>
<tr>
<td>5.7</td>
<td>Broadcast latency under different transmission radii</td>
<td>111</td>
</tr>
<tr>
<td>5.8</td>
<td>Transmission ratio under different transmission radii</td>
<td>112</td>
</tr>
<tr>
<td>6.1</td>
<td>A plot of $\rho$ when $\alpha = 4$ and $\beta = 1$</td>
<td>119</td>
</tr>
<tr>
<td>6.2</td>
<td>An example hexagonal tessellation and coloring</td>
<td>119</td>
</tr>
<tr>
<td>6.3</td>
<td>HBA in operation</td>
<td>124</td>
</tr>
</tbody>
</table>
6.4 An illustration of frame and sub-frame structure. Note, the sub-frame of (b) only applies to each slot of $\mathcal{F}^i(t)$ . . . . . . . . . . . . . . . 126
6.5 Broadcast latency under different number of nodes . . . . . . . . . 132
6.6 Number of transmissions under different number of nodes . . . . . 133
6.7 Broadcast latency under different range $r$ . . . . . . . . . . . . . . . 134
6.8 Number of transmissions under different range $r$ . . . . . . . . . . 134
6.9 Broadcast latency under different duty cycle . . . . . . . . . . . . . . 135
6.10 Number of transmissions under different duty cycle . . . . . . . . . 135
6.11 Performance of HBA versus Tree-based Algorithm . . . . . . . . . 137
# List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>A comparison of synchronous and asynchronous duty cycle.</td>
<td>3</td>
</tr>
<tr>
<td>1.2</td>
<td>Classification of interference models</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>List of contributions</td>
<td>10</td>
</tr>
<tr>
<td>2.1</td>
<td>Classification of neighbor knowledge methods</td>
<td>15</td>
</tr>
<tr>
<td>2.2</td>
<td>Classification of CDS methods</td>
<td>16</td>
</tr>
<tr>
<td>2.3</td>
<td>Comparison of various heuristics algorithms [75]</td>
<td>28</td>
</tr>
<tr>
<td>2.4</td>
<td>Comparison of MLBS algorithms</td>
<td>35</td>
</tr>
<tr>
<td>3.1</td>
<td>Definition of Notations in Chapter 3</td>
<td>41</td>
</tr>
<tr>
<td>3.2</td>
<td>Active time slots and layers of all nodes in Figure 3.1</td>
<td>43</td>
</tr>
<tr>
<td>4.1</td>
<td>Definition of notations used in Chapter 4</td>
<td>65</td>
</tr>
<tr>
<td>4.2</td>
<td>First three broadcast time slots of all nodes in Figure 4.1</td>
<td>68</td>
</tr>
<tr>
<td>5.1</td>
<td>Definition of Notations in Chapter 5</td>
<td>95</td>
</tr>
<tr>
<td>6.1</td>
<td>Definition of notations in Chapter 6</td>
<td>117</td>
</tr>
</tbody>
</table>
List of Abbreviations

WSN Wireless Sensor Network
MAC Medium Access Control
MLBS Minimum Latency Broadcast Scheduling
MLBSDC Minimum Latency Broadcast Scheduling for Duty-Cycled Networks
CDS Connected Dominating Set
RAD Random Access Delay
GPS Global Positioning System
UDG Unit Disc Graph
MPR Multipoint Relay
DS Dominating Set
CDS Connected Dominating Set
IS Independent Set
MIS Maximal Independent Set
MCDS Minimum CDS
BFS Breadth First Search
PBS Pipelined Broadcast Scheduling
CRN Cognitive Radio Network
MBS Mixed Broadcast Scheduling
SPT Shortest Path Tree
SINR Signal to Interference plus Noise Ratio
FTSP Flooding Time Synchronization Protocol
OPT Optimal Broadcast Algorithm
CEN Centralized Broadcast Algorithm
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIS</td>
<td>Distributed Broadcast Algorithm</td>
</tr>
<tr>
<td>MTS</td>
<td>Minimal Transmission Set</td>
</tr>
<tr>
<td>MITS</td>
<td>Maximal Independent Transmissions Set</td>
</tr>
<tr>
<td>MWIS</td>
<td>Maximal Weight Independent Set problem</td>
</tr>
<tr>
<td>PRG</td>
<td>Pseudo-Random number Generator</td>
</tr>
<tr>
<td>IABS</td>
<td>Interference-Aware Broadcast Scheduling</td>
</tr>
<tr>
<td>IABBS</td>
<td>Interference-Aware Basic Broadcast Scheduling</td>
</tr>
<tr>
<td>IAEBS</td>
<td>Interference-Aware Enhanced Broadcast Scheduling</td>
</tr>
<tr>
<td>HBA</td>
<td>Hexagon Broadcast Scheduling Algorithm</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

Wireless Sensor Networks (WSNs) consist of numerous sensor nodes deployed in a field [65]. They have been used in a variety of areas. Examples include monitoring an environment, health of machines and tracking targets in military applications [4] [12] [13] [82]. Sensor nodes are typically equipped with one or more radios, and communicate with each other via multi-hop, as these radios have a bounded and mostly short transmission range. Fig. 1.1 shows a sensor field with 13 sensor nodes, and a sink node connected to a laptop that is used to view sensed data. These sensor nodes are able to self organize and self-configure to form a connected network, and forward any sensed data to the sink node. The sink node is then responsible for processing said data and displaying them to users, which in turn may issue commands to effect the operation of sensor nodes.

Figure 1.1: A typical WSN architecture
1.1 Duty-Cycled Wireless Sensor Networks

A fundamental problem in WSNs is the finite battery capacity of each sensor node. This has a significant implication on the sensing rate of sensor nodes and protocol operation. As noted in [22], see Fig. 1.2, communication consumes the most energy in a wireless sensor node. In particular, the authors show that idle listening consumes the same energy as receiving. This thus motivated many researchers to develop duty-cycled Medium Access Control (MAC) protocols that control channel access, and more importantly nodes’ active and dormant states, whilst satisfying one or more application requirements [90].

![Figure 1.2: Power consumption of sub-systems on a sensor node [22].](image)

MAC protocols developed for WSNs can be classified into two families according to their clock synchronization requirement; namely, synchronous and asynchronous. The key characteristic of synchronous MAC protocols is the reliance on a global clock to ensure all nodes have the same time wake-up time. Example MACs in this family include S-MAC [90], T-MAC [77] and DMAC [54]. In contrast, nodes that use an asynchronous MAC protocol decide their wake-up schedule independently. Example MACs include B-MAC [64], Wise-MAC [21], X-MAC [11], RI-MAC [73] and PW-MAC [74]. The advantages and disadvantages of both categories are summarized in Table 1.1. For a comprehensive review of WSN MAC protocols, please refer to [17] [48] and references therein. As listed in Table 1.1, for WSNs that employ a synchronous schedule, all nodes have the same active time. However, nodes
will have to coordinate and synchronize their active time globally and hence, incur high signalling overheads. Therefore, this thesis only considers WSNs that employ an asynchronous MAC schedule, where nodes determine their active time independently and randomly.

Table 1.1
A comparison of synchronous and asynchronous duty cycle.

<table>
<thead>
<tr>
<th>Duty Cycle</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synchronous</td>
<td>• Energy efficient</td>
<td>• Complex synchronization algorithms</td>
</tr>
<tr>
<td></td>
<td>• Support broadcast efficiently</td>
<td>• Higher protocol overheads which leads to increased energy expenditure.</td>
</tr>
<tr>
<td>Asynchronous</td>
<td>• Simple</td>
<td>• Do not support broadcast efficiently.</td>
</tr>
<tr>
<td></td>
<td>• Energy efficient</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Collision avoidance property</td>
<td></td>
</tr>
</tbody>
</table>

Given the fact that sensor nodes wake up at different times with different active/sleep schedules, it is necessary to ensure that a sender is active at the same time as its intended receiver to transmit a message. To do this end, researchers have proposed three categories of asynchronous MAC protocols: preamble-based, receiver-initiated and duty-cycle-aware. In the preamble-based category, protocols such as [11] [21] and [64] require a sender to transmit a long preamble, which is usually longer than a receiver’s wake-up interval, before transmitting a message. Upon receiving a preamble message, a receiver will remain active and prepare to receive said message. However, preamble transmission occupies the wireless medium for a significant period of time, which increases delay and energy consumption. In the receiver-initiated category, [51] [73] [78], a sender wakes up and remains active until its intended receiver sends a beacon. That is, a node sends a beacon message whenever it wakes up. After receiving the beacon, the sender starts to transmit. In comparison to preamble-based methods, a receiver-initiated protocol avoids long preamble transmission, which decreases delay and energy consumption. However, the idle listening time of senders remain considerable. In the last category, a duty-cycle-aware protocol
such as [15] [50] and [74] is also referred to as pseudo-random protocol, where each node usually uses a pseudo-random sequence to generate its own active/sleep schedule independently. Therefore, a sender, who knows its intended receiver’s seed and last active time, can easily generate the receiver’s next active time. In order to maintain the active/sleep schedule information of neighboring nodes, nodes periodically send a beacon message including their ID, last wake-up time, and pseudo-random seed. The duty-cycle-aware protocol further reduces the energy consumption caused by idle listening.

Figure 1.3 illustrates the typical operation of asynchronous MACs. It shows node A, B and C have an independent active/sleep schedule, a key characteristic of asynchronous MACs. The duty cycle of a node is defined as the operation time of a sensor node or WSN. To be precise, duty cycle is the ratio between the active time and the scheduling period, denoted as $T$. As shown in Fig. 1.3, node A picks slot ‘2’ as its active time slot from a scheduling period $T$ of 4, and hence, its duty cycle is $\frac{1}{4}$.

![Diagram](image)

Figure 1.3: An example where node A, B and C adopt an asynchronous schedule

### 1.2 Broadcast

Network wide broadcast is a fundamental operation in wireless networks, where a message needs to be propagated from a source node, e.g., a sink, to all other nodes. It is relied upon by several network protocols, such as routing [63], information dissemination [59] and resource/services discovery [10]. For instance, many routing protocols, such as Ad Hoc On-Demand Distance Vector (AODV) [63], Dynamic Source Routing (DSR) [47] and Optimized Link State Routing protocol (OLSR) [43], rely on
broadcast mechanisms to disseminate local state information and control messages to all nodes in the network in order to establish routes between the source and destination. These protocols in turn help applications in disaster relief [2] [84] [85], military communication [1] [5], rescue operation [3] [9] and object detection [26] [70]. For these applications, time is critical, and hence, a minimum latency broadcast scheduling (MLBS) algorithm/protocol will be of great importance to their operation.

Like many other communication protocols, any developed MLBS solution must deal with interference which is one of the main performance-limiting factors in wireless networks [30] [89]. When two or more transmissions occur at the same time in the same frequency band, the signals from undesired or interfering transmitters are added to the desired transmitter’s signal, which prevent a receiver from decoding the transmitted signal correctly. Another problem when conducting broadcast is redundant transmissions. As mentioned, not all nodes in WSNs are within the transmission range of one another. Consequently, intermediate nodes are required to relay a broadcast message. These retransmissions take up valuable power and bandwidth resources. Moreover, there may be redundant retransmissions. Hence, it is important to choose intermediate nodes carefully so as to reduce redundant transmissions.

As shown in Figure 1.4, a broadcast requires source node $S$ to disseminate a message to nodes $A$, $B$, $C$ and $D$. In the first round, node $S$ sends the message to its one-hop neighbors successfully without any interference. After receiving the message, assume node $A$, $B$ and $C$ retransmit it to their common neighbor $D$ simultaneously.
in the second round. Due to collision, node $D$ will not receive any messages from $A$, $B$ or $C$ correctly. Moreover, since $D$ is a common neighbor of $A$, $B$ and $C$, it means two of the transmissions from $A$, $B$ and $C$ to $D$ are redundant; that is, only one transmission is sufficient in order to deliver the broadcast message to $D$. As a result of interference and redundant transmissions, node $D$ will not get the broadcast message successfully in the second round, which indicates more rounds or higher latency will be needed for source $S$ to finish the broadcast. Therefore, to reduce broadcast latency, it is necessary to address the problems of interference and redundant transmissions carefully.

Unfortunately, the MLBS problem for multi-hop wireless networks has been proven to be NP-hard [27], and researchers have proposed many approximation algorithms; see [14] [26] [27] [40] [41] [55] and [75]. To study MLBS problem, these approximation algorithms mainly assume three interference models; namely, $RTS/CTS$, $protocol$ and $physical$ $interference$ $model$. A summary of these interference models is listed in Table 1.2.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS/CTS</td>
<td>A node’s interference range is equal to its transmission range</td>
</tr>
<tr>
<td>Protocol</td>
<td>A node’s interference range is larger than its transmission range</td>
</tr>
<tr>
<td>Physical</td>
<td>The cumulative interference of many nodes outside the interference range is also taken into account</td>
</tr>
</tbody>
</table>

To date, a majority of solutions to the MLBS problem assume the RTS/CTS model or protocol interference model [36]. Note, the RTS/CTS interference model and protocol interference model are also together referred to as bounded interference models. As illustrated in Figure 1.5, these works usually adopt highly theoretical disk graph models, in which the transmission range $r_T$ and interference range $r_I$ is thought of as a disk centred at a node. The RTS/CTS interference model only considers the collisions within a node’s transmission range; that is, interference and transmission range
are equal ($r_T = r_I$). On the other hand, the protocol interference model assumes the interference range is larger than transmission range ($r_T < r_I$). Specifically, nodes that adopt this model assume there is interference when nodes lie in the overlapping region within their interference range. These ‘interfered nodes’ must therefore be scheduled in different time slots according to topological information.

![Bounded interference model](image)

**Figure 1.5: Bounded interference model**

A key limitation of bounded interference or RTS/CTS models is that they cannot model the case where many far-away nodes could still have a non-negligible effect on reception. To this end, the physical interference model, also called SINR-based interference model, is more realistic, where the cumulative interference of many nodes outside the interference range are not neglected. To date, only a few broadcast algorithms [39] [40] [81] have been designed for the MLBS problem under the physical interference model; see Chapter 2.

### 1.3 Problem Space and Motivation

The MLBS problem is quite different in duty-cycled WSNs. Briefly, any solutions to the MLBS problem in duty-cycled WSNs (MLBSDC) will have to consider both topological information and active/sleep or wake-up schedules of nodes to avoid interference and to ensure prompt delivery. It is because multi-hop broadcast is not efficiently supported by existing asynchronous duty-cycled MACs, i.e., nodes may have to transmit a broadcast message multiple times because each neighbor has its
own wake-up schedule. This problem is further exacerbated when broadcasting over multiple hops. In particular, nodes may interfere with one another.

As an example, consider Figure 1.4. Node $S$ needs to broadcast a message to node $A$, $B$, $C$ and $D$. Assume all of them have a different wake-up time, i.e., time slot ‘1’, ‘3’, ‘5’ and ‘5’, respectively. Here, node $S$ may transmit the message at least three times because its neighbors $A$, $B$ and $C$ have a different wake-up time. Moreover, assuming node $A$ has received the message from $S$ at time slot ‘1’ and $B$ received the message from $S$ at time slot ‘3’, node $S$, $A$ and $B$ may try to forward the message to their neighbors at time slot ‘5’. However, this will cause a collision at node $C$ and $D$. Considering the fact that $B$ is adjacent to node $C$ and $D$, both with the same wake-up time of ‘5’, one feasible way to conduct the broadcast is for $S$ to send the message to $A$ and $B$ at time slot ‘1’ and ‘3’, respectively, after which $B$ transmits it to $C$ and $D$ at time slot ‘5’. As we can see, both topology and active/sleep or wake-up schedule of nodes are key issues to consider when solving the MLBSDC problem. In fact, this consideration renders the MLBS problem more complex, meaning existing algorithms for always-on wireless networks are no longer applicable. To date, only a handful of papers [38] [46] [88] have tried to address the MLBS problem in duty-cycled wireless networks, and critically, all these works have only studied the problem under the RTS/CTS interference model.

1.4 Contributions

This thesis, therefore, aims to study the MLBSDC problem under three different interference models, i.e., RTC/CTS, protocol, and physical interference. Specifically, it designs and evaluates centralized and distributed broadcast algorithms with minimum latency. As listed in Table 1.3, the objectives of this thesis include: (1) to design centralized and distributed collision-free broadcast algorithms with minimum latency for duty-cycled WSNs based on the RTS/CTS interference model, (2) to design interference-free broadcast algorithms for always-on wireless networks and duty-cycled WSNs, which improve upon current state of the art performance in terms of broadcast latency and redundant transmissions under the protocol interference model.
model, and (3) to broadcast messages with low latency and redundant transmissions over the physical interference model. The contributions are listed as follows:

1. This thesis presents two approximation algorithms, called BS-1 and BS-2 respectively, for the MLBSDC problem under the RTS/CTS interference model. These algorithms produce a broadcast schedule with a ratio of $(\Delta - 1)T$ and $13T$ for latency, respectively. Here, $\Delta$ is the maximum degree of nodes, and $T$ denotes the number of time slots in a scheduling period. To date, BS-2 produces the best constant approximation ratio of $13T$ as compared to other proposed algorithms; viz. $24T + 1$ and $17T$ for the algorithm reported in [38] and [26] respectively. This thesis also proves that the total number of transmissions produced by BS-2 is at most $4(T + 3)$ times larger than that of the minimum total number of transmissions. Extensive experimental results show that on average, the proposed algorithms have a near optimal performance in terms of broadcast latency. In particular, compared to [26], the best broadcast scheduling algorithm to date, the broadcast latency and number of transmission achieved by BS-1 is at least $1/17$ and $2/5$ of that of [26] respectively.

2. This thesis proposes a centralized (CEN) and distributed (DIS) greedy heuristic algorithm for the MLBSDC problem under the RTS/CTS interference model. CEN produces a ratio of $(\Delta - 1)T$ in terms of broadcast latency. DIS is a distributed implementation of CEN based on the local information of a node’s two-hop neighbors. Extensive experimental results show that the broadcast latency of CEN and DIS is near optimal. In particular, compared to [26], the broadcast latency and transmission times achieved by CEN are about $1/5$ and $1/2$ of that of [26], respectively.

3. This thesis outlines two constant approximation algorithms, called IABBS and IAEBS respectively, for the MLBS problem under the protocol interference model. These algorithms produce a broadcast schedule with a ratio of $2 \left\lceil \frac{\pi}{\sqrt{3}} (\rho + 1)^2 + \left( \frac{\pi}{2} + 1 \right)(\rho + 1) + 1 \right\rceil$ with respect to broadcast latency, where $\rho = \frac{r_I}{r_T}$. It also proves that the total number of transmissions produced by IABBS and IAEBS is at most eight times larger than the minimum total number of transmissions. It confirms that the latencies achieved by IABBS and
Table 1.3
List of contributions

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Types</th>
<th>Groups</th>
<th>Interference models</th>
<th>Networks</th>
<th>Approximation ratio (^a)</th>
<th>Approximation ratio (^b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BS-1</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>((\Delta - 1)T)</td>
<td>-</td>
</tr>
<tr>
<td>BS-2</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>(13T)</td>
<td>(4T + 12)</td>
</tr>
<tr>
<td>CEN and DIS</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>((\Delta - 1)T)</td>
<td>-</td>
</tr>
<tr>
<td>IABBS</td>
<td>Centralised</td>
<td>Greedy heuristic</td>
<td>Protocol</td>
<td>Always-on</td>
<td>(2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor T)</td>
<td>(8T)</td>
</tr>
<tr>
<td>IAEBS</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>Protocol</td>
<td>Always-on</td>
<td>(2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor T)</td>
<td>(8T)</td>
</tr>
<tr>
<td>HBA</td>
<td>Distributed</td>
<td>Greedy heuristic</td>
<td>Physical</td>
<td>Duty-cycled</td>
<td>(9 \left\lfloor 2 + \frac{2(8.9)}{1 - (r/r_{\text{max}})^\alpha} \left(\frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3\right)^{1/\alpha} \right\rfloor T)</td>
<td>-</td>
</tr>
</tbody>
</table>

\(^a\)This approximation ratio is with regards to broadcast latency

\(^b\)This approximation ratio is with respect to the number of transmissions
IAEBS are much lower than existing schemes. In particular, compared to CABS [55], the best constant approximation broadcast algorithm to date, the broadcast latency achieved by IAEBS is only \( \frac{3}{8} \) of that of CABS. The thesis also shows that both IABBS and IAEBS are also suitable for duty-cycled WSNs.

4. This thesis is the first to design a distributed broadcast algorithm, called HBA, for the MLBSDC problem under the physical interference model. For broadcast latency, HBA produces a constant approximation ratio of

\[
9 \left[ 2 + \frac{2}{3} \left( 1 - \left( \frac{r}{r_{\text{max}}} \right)^\alpha \left( \frac{2}{\alpha - 2} + \frac{1}{\alpha - 1} + 3 \right) \right)^{1/\alpha} \right]^2 T
\]

where \( \alpha \) is the path-loss exponent, \( \beta \) is the minimum SINR threshold required for a message to be decoded successfully, \( r_{\text{max}} \) is the maximum transmission range, and \( r \) is nodes’ transmission range. The total number of transmissions in terms of broadcast messages produced by HBA is upper-bounded by \((T + 1)N_H\), where \( N_H \) is the number of hexagons required to cover the entire network. Extensive experimental results show that on average, HBA has a much better performance, i.e., \( \frac{1}{2} \), in terms of broadcast latency than the tree-based Algorithm [40]. The key reason is because HBA is able to schedule transmissions in multiple layers as opposed to layer by layer, as is done by [40].

1.5 Publications

This thesis has resulted in the following papers:


### 1.6 Thesis Structure

1. *Chapter 2*. This chapter includes a literature review of existing approaches on MLBS problem in always-on wireless networks and duty-cycled WSNs.

2. *Chapter 3*. This chapter introduces two approximation approaches, called BS-1 and BS-2 respectively, on MLBSDC problem under the RTS/CTS interference model.

3. *Chapter 4*. This chapter proposes a centralised algorithm and its distributed implementation, namely CEN/DIS, on MLBSDC problem under the RTS/CTS interference model.

4. *Chapter 5*. This chapter outlines two constant approximation approaches, called IABBS and IAEBS respectively, for MLBS problem under the protocol interference model.

5. *Chapter 6*. This chapter designs a distributed constant approximation approach, called HAB, on MLBSDC problem under the physical interference model.

6. *Chapter 7*. This chapter concludes the thesis, and provides a summary of research outcomes and future research directions.
Literature Review

A lot of studies have tried to address the broadcast problem. One of the earliest studies uses flooding [53], where each node retransmits a broadcast message after receiving it. Although flooding is simple and easy to implement, it suffers from the well-known broadcast storm problem [60] [76], where nodes experience an exorbitant amount of redundant transmissions, bandwidth contention and collision. Consequently, a number of researchers, e.g., [23] [68] [71], have proposed methods that improve the efficiency of broadcast, i.e., avoiding interference and reducing redundant transmissions. In the ensuing sections, existing works related to MLBS algorithms for always-on wireless networks are first discussed. After that, Section 2.3 discusses broadcast algorithms for duty-cycled WSNs.

2.1 Broadcast over Always-on Wireless Networks

Most existing broadcast protocols try to reduce the probability of interferences by limiting the number of transmissions in the network. In general, these broadcast protocols are categorized into three groups: probability, area and neighbor knowledge based methods [86].
2.1.1 Probability Based Methods

In probability based methods, nodes need to rebroadcast with a predetermined probability $p$. As shown in [92], $p$ is proportional to a node’s degree. For example, a node with a high degree or immediate neighbors will be assigned a lower $p$ value, while a node will be assigned a higher $p$ value otherwise. The intuition here is that nodes with a higher degree are likely to have overlapping coverage areas, and hence using a lower $p$ leads to a lower number of transmissions. The probability also relies on the number of times a broadcast message is received at a node. Ni et al. [60] assign each node a random Assessment Delay (RAD), which is a random interval of time used by nodes to listen to the wireless channel. Upon receiving a duplicated broadcast message, a node increases its counter by one. After RAD time, the node evaluates its counter and only rebroadcasts when the counter is less than a given threshold. These methods may work in dense networks where nodes have multiple overlapping coverage areas, but will not have a significant effect in sparse networks.

2.1.2 Area Based Methods

In the area based methods, the given criterion is the distance between neighboring nodes. The assumption is that if the distance between two neighboring nodes is very small, the additional coverage area provided by a neighboring node will tend to be small. A node rebroadcasts only when it receives a message from a node that is beyond a given distance threshold, so that a larger additional coverage area can be reached. For instance, in [92], if a node receives a broadcast message for the first time, a RAD will be activated. Before the RAD expires, it records all redundant messages and the distance of each sender. If any sender is outside the given distance threshold, a node waits for RAD time before rebroadcasting messages.

Nodes can also calculate a more precise additional coverage area by using the Global Positioning System (GPS) to estimate their location. Durresi et al. [19] propose an optimized broadcast protocol for sensor networks. Their protocol divides a sensing area into numerous hexagons with radius set to nodes’ transmission range. If a node rebroadcasts a message, it will add its location into the header of the message, as well as the location of the node from which it received the message. Upon receiving
a broadcast message, the receiver will calculate the distance from the sender using the location information. If the distance is less than a threshold value, the receiver will not rebroadcast. Otherwise, the node will set a RAD based on the distance from the nearest hexagon vertex. This RAD indicates that nodes closest to a hexagon vertex will rebroadcast first with the smallest delay. Before RAD expires, the node will collect redundant messages and again examine its nearest distance from senders. If it is within the threshold value, it will not rebroadcast. However, the area based methods do not consider whether nodes actually exist within the additional coverage area, incurring more redundant transmissions.

2.1.3 Neighbor Knowledge Methods

In neighbor knowledge methods, the key characteristic is to construct a rebroadcast set, where only nodes belonging to this set are allowed to transmit, using local topology information, e.g., one-hop neighbors. These methods can be further classified into two groups: source dependent and source independent methods. As shown in Table 2.1, in the source dependent methods, when a node transmits a message, it needs to specify which of its one-hop neighbors are allowed to retransmit the message; instead, in the source independent methods, upon receiving a message, a node will decide whether or not to retransmit the message by itself.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Dependent Methods</td>
<td>When a node sends a message, it specifies which of its one-hop neighbors need to retransmit the message</td>
</tr>
<tr>
<td>Source Independent Methods</td>
<td>Upon receiving a message, a node decides whether or not to retransmit the message by itself</td>
</tr>
</tbody>
</table>

An example of source dependent method is Multipoint Relay (MPR) [66]. A sender selects a subset of its one-hop neighbors, called MPRs, to rebroadcast any messages. Moreover, only nodes belonging to MPRs are allowed to rebroadcast a message. The
MPR set is constructed as follows. A sender $v$ repeatedly selects into the MPR set a node $u$ that covers the most two-hop neighbors of $v$ that have yet to be covered by any nodes already in the MPR set. Then, it removes $u$ and covered two-hop neighbors from $v$’s two-hop neighborhood. The said steps are repeated until all of $v$’s two-hop neighbors are covered. In the implementation, a chosen MPR node $v$ broadcasts a HELLO message that includes its chosen MPRs. Upon receiving a HELLO message, a neighbor checks whether it belongs to the MPR set. If so, it must rebroadcast any messages received from $v$.

Kim et al. [52] proposed a simple source independent method called self pruning. Nodes add a neighbors’ list into each broadcast message. Upon receiving a message, a neighbor examines to see whether there are nodes that have yet to receive the broadcast message. If there are no such nodes, it stays silent. Otherwise, it rebroadcasts the message.

Many proposed algorithms are based on constructing a connected dominating set (CDS), where only nodes in CDS are allowed to retransmit a broadcast message. For a graph $G$, a CDS is defined as a subset of $G$ such that the subgraph induced by CDS is connected and each vertex in $G$ is either within CDS or adjacent to a vertex in CDS. That is, CDS is a dominating set (DS) of $G$. However, finding a minimum CDS (MCDS) in a network has been shown to be NP-hard [28], and therefore, many approximation algorithms are presented. Generally, exiting CDS algorithms can be divided into three groups [8]: tree, cluster and pruning based methods; see Table 2.2.

<table>
<thead>
<tr>
<th>Groups</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tree</td>
<td>Grow a spanning tree, and only non-leaf nodes constructs a CDS</td>
</tr>
<tr>
<td>Cluster</td>
<td>Construct a dominating set (DS) first, and then connect nodes in DS by picking connectors</td>
</tr>
<tr>
<td>Prunning</td>
<td>Construct a coarse CDS initially, and then reduce its size</td>
</tr>
</tbody>
</table>
In tree-based CDS methods, they need to grow a spanning tree, and only non-leaf nodes in this tree will construct a CDS. Guha et al. [34] described a simple tree-based algorithm. Initially, all nodes in the network are marked white. The algorithm then selects a white node with the maximum degree as the root, and marks it as black. Then, it repeats the following steps to extend the tree until all nodes are visited. First, it includes one-hop neighbors of black nodes and colors them gray. Next, it selects a pair of gray nodes and their neighboring white nodes that have the most white neighbors, and mark them black. Finally, all black nodes or non-leaf nodes in the tree construct a CDS.

Cluster-based methods start by constructing a dominating set (DS), and then connect the nodes in this set by choosing connectors, which are chosen from nodes outside a DS and used to connect nodes in DS. In [34], Guha et al. also proposed a cluster-based algorithm. Nodes in DS are called black nodes, and gray nodes are those that do not belong to a DS but are covered by a DS. White nodes are neither in DS nor covered by DS. They define a piece as a white node or a connected subgraph induced by black nodes. Then, their method operates in two phases. In the first phase, it recursively selects into the DS set a node that reduces the maximum number of pieces. They show at the end of this phase no white nodes will be left when the number of pieces cannot be reduced further. As a result, black nodes form a DS, which consists of several separate clusters or CDSs. Next, their algorithm selects a pair of nodes as connectors to link separate CDSs until all separate CDSs form a connected dominating set.

The pruning-based methods first construct a coarse CDS, and then reduce its size. One typical pruning-based CDS algorithm is proposed by Wu et al. [87]. Each node uses its two-hop neighborhood information to choose a set of dominators. Specifically, a node with at least two unconnected neighbors will be selected as a dominator. They then prove all selected dominators will form a CDS. Then, they presented two rules to remove redundant dominators from the constructed CDS. A dominator $v$ is removed if $v$ and its neighbors are covered by another dominator $u$, and $v$’s ID is smaller than $u$’s ID. Secondly, a dominator $v$ is removed if $v$ and its neighbors are covered by two other dominators $u$ and $w$, and $v$ has the smallest ID.
2.2 MLBS over Always-on Wireless Networks

Thus far, the aforementioned works are not able to avoid all interference; that is they cannot ensure every node receives a broadcast message interfere-free. Furthermore, none of these works guarantee any bounds on broadcast latency. Nevertheless, MLBS aims to find an efficient, interference-free schedule that yields the minimum broadcast latency and redundancy of transmissions. This problem, MLBS, has been studied extensively in wireless multi-hop networks whereby nodes remain on all the time, and has been shown to be NP-hard [27]. Researchers have proposed many approximation algorithms. In general, these existing MLBS related works can be categorized into two groups: tree and greedy heuristics.

2.2.1 Tree Based Algorithms

Most studies have addressed the MLBS problem by constructing a broadcast tree rooted at the source node, in which a parent node is responsible for relaying a broadcast message to all of its children without any interference. This ensures a deterministic approximation ratio in terms of broadcast latency and the number of transmissions.

Chlanitac et al. [16] present an approximation algorithm with a broadcast latency of $O(\Delta R)$ over the RTS/CTS interference model, where $\Delta$ is the maximum degree of nodes and $R$ is the maximum Euclidean hop distance from the source to any node. They schedule the broadcast based on a breadth first search (BFS) tree rooted at the source node. This BFS tree is divided into different layers, where each layer contains nodes with the same minimum hop-distance to the source. Specifically, their scheduling is conducted layer by layer in a top-down manner. For each layer $i$, this algorithm randomly selects a node $v$ from this layer, and assigns it as the parent of its one-hop neighbors that do not yet have a parent. Here, a parent is responsible for retransmitting a broadcast message to its children. Then, this algorithm allocates parent $v$ with a minimal transmission time $t$ based on the following collision-free condition: no nodes transmit at $t$ among $v$’s two-hop neighborhood. The above steps are repeated until all nodes in layer $i$ are considered. Other layers in the BFS are
scheduled in a similar manner. They prove their algorithm produces a broadcast latency of $O(\Delta R)$. However, the number of transmissions $n$, where $n$ is the total number of nodes.

Gandhi et al. [27] presented an algorithm with a constant approximation ratio of more than 400 with respect to broadcast latency. Their work also assumes the RTS/CTS interference model. Their algorithm constructs a broadcast tree in two steps. At first, they build a maximum independent set (MIS) by iterating the BFS in a top down manner. MIS is also called independent dominating set which means no vertices in MIS are adjacent, and every vertex not in MIS is joined to at least one member of MIS by some edge for a graph $G$. Specifically, they select nodes at each layer of the BFS that do not have neighbors in the MIS greedily. In the second step, for each layer $i$ of the BFS, they choose nodes in (respectively, outside) MIS as the parent of nodes outside (respectively, in) MIS in an arbitrary order. Specifically, a node belonging to MIS in layer $i$ will be chosen as the parent of nodes outside MIS in layer $i$ and $i + 1$; while a node outside MIS in layer $i$ will become a node in MIS lying in layer $i + 1$.

The broadcast tree is constructed after all nodes are assigned a parent. Then, they adopt a greedy collision-free method to schedule the transmissions of parent nodes. For each parent $v$, they assign $v$ with a minimum transmission time $t$ satisfying the following conditions: 1) $v$ receives the broadcast message from its parent before $t$; 2) no $v$’s children hear any transmissions at $t$; 3) no $v$’s neighbors that do not belong to $v$’s children are receiving from their parent nodes at $t$. They prove their algorithm produces a constant approximation ratio in terms of broadcast latency and number of transmissions.

Huang et al. [41] propose an algorithm, assuming the RTS/CTS interference model, with a broadcast latency of at most $24R$. They construct a broadcast tree in the same way as [27]. The nodes in MIS are referred to as dominators, and the parents of dominators are also referred to as connectors. Dominators together with connectors form a CDS, and only dominators and connectors are allowed to transmit. The transmissions of dominators and connectors are scheduled layer by layer in a top-down manner based on the BFS. At each layer, they mark dominators with a different color if they share the same neighbors, which mean dominators labelled with a different
color will result in a collision if they transmit or receive simultaneously. For each layer $i$, the transmissions of dominators are first scheduled based on their colors; that is, dominators with a different color will be assigned non-interfering transmission times. Then, the transmissions of connectors in layer $i$ are scheduled in a similar way based on the color of their children, which are the dominators in layer $i + 1$.

To further reduce the broadcast latency produced by the first algorithm in [41], Huang et al. [41] outline an algorithm that yields a broadcast latency of at most $16R$. Different from the first algorithm, after constructing the MIS, the proposed algorithm preferentially selects as a connector a node that covers the most dominators that have yet to be assigned a parent (connectors in the first algorithm are chosen in an arbitrary manner). Then, the transmissions of dominators in layer $i$ are first scheduled based on their color. However, the transmissions of connectors in layer $i$ will be scheduled based on their own color (in the first algorithm, the schedule is based on the color of a connector’s neighboring dominators). It means connectors will be allocated with a different color if they are adjacent to the same dominators in layer $i + 1$.

The main drawback of the first and second algorithms described in [41] is that all nodes in a BFS layer must be informed before the broadcast proceeds to subsequent layers. In other words, transmissions in subsequent layers that do not cause interference will be blocked until nodes in upper layers have finished their reception. Consequently, this increases broadcast latency unnecessarily. To address this problem, Huang et al. [41] present a third algorithm called pipelined broadcast scheduling (PBS) with a latency of at most $R + O(\log_2 R)$. Instead of scheduling transmissions layer by layer in a top-down manner, PBS schedules simultaneous transmissions in more than one layer. PBS starts by constructing a ranking tree, whereby nodes with the greatest rank will be scheduled to transmit first. PBS assigns a rank to each node in the BFS tree rooted at the source node layer by layer in a bottom-up manner. Basically, if a node from layer $i$ covers most nodes in lower layer $i + 1$ with the same rank $r$, PBS will update this node’s rank to $r + 1$, and selects this node as the parent to cover nodes in layer $i + 1$. In [41], the authors show the source node will be assigned with the largest rank, say $r_0$. Then, a pipelined session is defined as transmissions from nodes in layer $i$ with rank $r$ to their children that are in layer

Each pipelined session starts to transmit at \( i + 51(r_0 - r) \). To avoid collisions in a pipelined session, PBS marks parent nodes with a different color if they have a common neighbor, and then assigns them with different transmission times based on their color. For instance, if a parent node is assigned with a color label \( k \), its transmission time will be set to \( i + 51(r_0 - r) + 3k \), where ‘3’ is used to interleave transmissions from other layers \( i', |i' - i| \leq 2 \). Finally, they show the largest transmission time in PBS is bounded by \( R + O(\log_2 R) \). However, the omitted constant in \( O(\log_2 R) \) exceeds 150.

Gandhi et al. [26] improved the broadcast latency approximation ratio of [27] from 400 to 12; to date, this is the best ratio for works that assume the RTS/CTS interference model. Similar to [27], they initially construct a MIS set in a top down manner of the BFS. In [26], nodes in MIS are called primary, and their parents are referred to as secondary nodes. A primary node in layer \( i \) of the BFS is selected as the parent of nodes not belonging to MIS in layer \( i \) or \( i + 1 \) if said primary node covers the most nodes that do not yet have a parent. A secondary node in layer \( i \) is chosen from nodes not belonging to the MIS and designated as the parent of primary nodes in layer \( i + 1 \) in the same way. Note, parent nodes in [27] are chosen arbitrarily. Primary together with secondary nodes form a CDS. Only nodes in CDS are allowed to retransmit a broadcast message. The transmissions of nodes in CDS are scheduled in two phases. In phase 1, only nodes in CDS are scheduled to transmit the broadcast message. In phase 2, nodes in CDS transmit the message to all other nodes. In both phases, they allocate parent nodes with a minimal transmitting time in the same way as [27]. That is, at this minimal transmitting time, simultaneous transmissions must be collision-free within a parent node’s two-hop neighborhood. Finally, Gandhi et al. [26] prove their algorithm yields a 12-approximate solution for latency, and the number of transmissions is at most 21 times larger than that of an optimal solution.

Ji et al. [44] study the MLBS problem under the RTS/CTS interference model for Cognitive Radio Networks (CRNs) and propose a Mixed Broadcast Scheduling algorithm (MBS). CRN consists of licensed and unlicensed users. Here, unlicensed users must coexist with licensed users; that is, an unlicensed user needs to sense and learn its wireless channel before starting a data transmission. If the channel is idle,
this unlicensed user will start its data transmission. Hence, the MLBS problem in CRNS not only requires the transmissions between unlicensed users to be collision-free, but any solutions must also ensure that there is no collision between unlicensed and licensed users. To this end, MBS starts by constructing a CDS set based on the BFS tree rooted at the source using the same method in [26]. It then schedules the transmissions of nodes in CDS in two phases. In the first phase, a broadcast message is sent to all nodes in the CDS via unicast transmissions. In the second phase, it adopts the hexagonal coloring technique introduced in [40] to partition the dominators in CDS into different subsets, where dominators in the same subset are allowed to transmit simultaneously. Then, dominators in CDS send the broadcast message to all other unlicensed users. Furthermore, to avoid collisions between unlicensed users and licensed users, MBS requires each unlicensed user to sense the channel randomly before retransmitting a broadcast message.

Chen et al. [14] are the first to study the MLBS problem under the protocol interference model, where the interference range is larger than the transmission range. In [14], they proposed an algorithm called interference aware broadcast (IAB) which produces a constant approximation ratio for broadcast latency. IAB starts by constructing a BFS tree rooted at the source node. To accelerate transmissions, IAB performs two types of transmissions: forward and backward. The former are defined as transmissions from nodes in layer $i$ to layer $i + 1$. The latter transmissions are processed from nodes in layer $i$ to nodes in layer $i$ or $i - 1$. Instead of scheduling transmissions in a top-down manner, that is, forward transmissions, IAB uses backward transmissions to allow more interference-free transmissions in more than one layer. Specifically, at each layer of the BFS, rather than covering all nodes in layer $i + 1$ through forward transmissions, IAB only assigns $\delta$ time slots for forward transmissions to reach nodes in layer $i + 1$, where $\delta = 26$, assuming an interference range that is two times that of the transmission range. After $\delta$ time slots, if there are any uncovered nodes in layer $i + 1$, IAB uses backward transmissions. At each time slot, a maximal interference free set is computed to schedule forward and backward transmissions. Namely, IAB recursively selects into this maximal interference-free set some senders in forward and backward transmissions that will not interfere with each other. That is, when a sender transmits a message, there are no other senders
within its receiver’s interference range transmitting at the same time. They prove IAB gives a constant approximation algorithm of 26 in terms of broadcast latency.

Huang et al. [40] proposed an algorithm with a ratio of $6\left\lceil \frac{2}{3} (\rho + 2) \right\rceil^2$ under the protocol interference model, where $\rho = \frac{r_I}{r_T}$, $r_I$ and $r_T$ denote the interference and transmission range respectively. They initially partition a network field into equal hexagons with a radius of $\frac{r_T}{2}$. Then, they mark hexagons with a different color if their mutual distance is less than $r_T + r_I$. They show that in order for simultaneous transmissions to be interference-free in the protocol interference model, it is sufficient to have the condition that the mutual distance between senders or receivers is larger than $r_T + r_I$. To provide a low-latency solution, their proposed algorithm involves two steps: constructing a broadcast tree and scheduling the transmissions of parent nodes. In the first step, they start by finding a layered MIS set based on the BFS tree rooted at the source node. Then, they choose a node from layer $i$ as the parent of dominators in MIS in layer $i$, also referred to as connectors. Similarly, a dominator in layer $i$ will also be chosen as the parent of nodes in layer $i$ and $i + 1$. In the second step, at each layer $i$ of the BFS, they first allocate connectors in layer $i - 1$ with a transmission time based on the color of hexagon of their children. Then, they assign dominators in layer $i$ with a transmission time based on the color of their hexagon.

Tiwari et al. [75] extend Huang et al.’s method [40] to consider different transmission ranges and dimensions, i.e., 2D and 3D, and presented an approximation algorithm with a constant ratio of $2 \left\lceil \frac{4}{3} (\rho + 1)^2 \chi^2 + \frac{8}{3} (\rho + 1) \chi + \frac{4}{3} \right\rceil$ for the 2D space, where $\chi$ is ratio of the maximum and minimum transmission range. They construct a broadcast tree rooted at the source node in the same way as [40]. A key difference is that they employ a more efficient method to color hexagons; i.e., one that produces fewer colors. Therefore, their algorithm requires fewer transmission time slots. Moreover, they produce the first distributed algorithm for MLBS problem under the protocol interference model. In their distributed implementation, for each hexagon, they iteratively construct a covering set to cover nodes outside the hexagon. Basically, they preferentially select into the covering set a node that covers the most nodes that have yet to be covered by any nodes already in the covering set. They call nodes in the
covering set as suppliers and their covered nodes as providers, where a provider is responsible for transmitting a message to its corresponding supplier. In [75], time is divided into different frames, which are equal to the number of colors used by hexagons. These frames are then allocated to hexagons marked with the corresponding color. Next, for each hexagon, at its corresponding frame, a supplier node will broadcast a REQUEST message to all of its providers. If any of these providers have received a broadcast message before, they will reply with a RESPONSE Message. Upon receiving the first RESPONSE message successfully, a supplier responds with a RECEIVING message to indicate to the corresponding provider to start transmitting. After receiving the broadcast message from its provider, the supplier broadcasts this message to its neighbors. However, the key limitation is that a supplier needs to exchange a significant number of control messages with its providers until it receives the broadcast message. This limitation leads to an increase in bandwidth contention and energy consumption.

Mahjourian et al. [55] study the conflict-aware MLBS problem under the protocol interference model whereby apart from the transmission and interference range, they also consider the carrier sensing range. They propose a constant approximation algorithm called conflict-aware broadcast scheduling (CABS) that has a ratio of $O(\min(\frac{r_T}{r_I}, \frac{r_S}{r_T})^2)$, where $r_S$ is the carrier sensing range. The mechanism they use to avoid conflicts is that in order for two simultaneous transmissions to be conflict-free, it is sufficient to have the mutual distance between senders too be larger than $\max(r_T + r_I, r_S)$, or the mutual distance between two receivers to be larger than $\max(r_I, r_S) + 2r_T$. However, these two conditions are in general stronger than what is needed for avoiding conflicts, which means, two simultaneous transmissions may be conflict-free, even if they do not satisfy these conditions. CABS then uses these two conditions to construct two conflict graphs: transmission and reception. Namely, in the transmission conflict graph, an edge exists between two nodes if their mutual distance is no larger than $\max(r_T + r_I, r_S)$. Similarly, in the reception conflict graph, two nodes are connected by an edge if they violate the second conflict-free condition. CABS constructs a broadcast tree in the same way as [40]. A dominator in layer $i$ of a BFS tree rooted at the source will be the parent of nodes in layer $i$ and $i + 1$; while a node not belonging to MIS in layer $i$ will be chosen as the parent of dominators
in layer $i + 1$. Next, at each layer $i$, transmissions by dominators will be scheduled based on the transmission conflict graph. In particular, dominators in layer $i$ will be assigned a different transmission time if they are adjacent in the transmission conflict graph. Likewise, nodes in layer $i$ that are the parent of dominators in layer $i + 1$ will be allocated a different transmission time if their children (dominators in layer $i + 1$) are neighbors in the reception conflict graph.

There are only a handful of works that have considered the MLBS problem over the physical interference model. Huang et. al [40] propose the first approximation algorithm based on the observation that if the mutual distance between two senders or receivers is larger than a threshold $r_i$, two simultaneous transmissions can be interference-free even under the physical interference model. They then prove this threshold $r_i$ is related to a node’s transmission power $P$, background noise $N$, path loss exponent $\alpha$ and the minimum signal to interference plus noise ratio (SINR) $\beta$. Using this observation, they partition the network into equal hexagons with a radius of $\frac{r_i}{2}$, where $r_i$ is a range that is no larger than nodes’ maximum transmission range. This is carried out to ensure connectivity. Then, they mark hexagons with a different color if these hexagons are separated by a distance less than threshold $r_i$. Next, they extend their algorithm in [40], which was developed for the protocol interference model, to consider the physical interference model. They construct a broadcast tree and assign parent nodes with interference-free transmission times based on hexagons’ color. Finally, they prove their method is a $O((\frac{r_i}{r_t})^2)$-approximate solution for the MLBS problem under the physical interference model.

Huang et al. [39] then extend the approach in [40] to consider a more realistic interference model, where a message is received successfully with a given probability should the receiving SINR falls below $\beta$, which is the minimum SINR threshold required for a message to be received successfully. In their algorithm, each parent node is required to repeat transmitting a message multiple times to ensure its children can receive the message successfully.

Wan et al. [81] propose an algorithm for the MLBS problem under the physical interference model, which has a constant approximation ratio for broadcast latency. Initially, they convert the network under the physical interference model into a disk
graph. To be specific, similar to [40], they assign each node with a communication range $r_t$ that is no larger than the maximum transmission range, and guarantees connectivity in the resulting disk graph. They also define a distance threshold $r_i$ smaller than that in [40], where if the mutual distance between senders or receivers is larger than $r_i$, then they assume their transmissions are interference-free. The next step is to construct a CDS based on the converted disk graph, which follows a two-phase method. The first phase is used to construct a MIS induced by the increasing order of BFS rooted at the source. The second phase constructs a set of connectors, which together with the dominators in the MIS form a CDS. Specifically, a subgraph $G'$ of dominators is first constructed in which an edge exists between two dominators if they have a common neighbor. Then, for each layer $i$ of the BFS tree in $G'$ rooted at the source, they preferentially select as the connector a node not belonging to the MIS and covers the most dominators in layer $i$ and $i+1$. To schedule interference-free transmissions, they compute a distance-$r_i$ coloring for dominators in each layer of the BFS of $G'$ such that two dominators with a mutual distance less than $r_i$ are assigned a different color. In their scheduling, a dominator’s transmission time is calculated based on its own color. However, the transmission time of a connector is computed according to its children’s color.

### 2.2.2 Greedy Heuristic Algorithms

As was mentioned above, most existing tree-based algorithms for the MLBS problem require all nodes in a BFS layer to be informed before the broadcast proceeds to subsequent layers. This mechanism will prevent interference-free transmissions from subsequent layers, and give rise to high latency. To avoid this layer-by-layer approach, greedy heuristic algorithms usually consider the set of all nodes that have received the broadcast message as potential transmitters. Consequently, from the view of the broadcast tree, greedy heuristic algorithms are able to schedule simultaneous transmissions from multiple layers, thereby, speeding up the propagation of a broadcast message. Therefore, in each time slot, greedy heuristic algorithms will select a maximal subset of interference-free transmitters based on an interference avoidance technique. That is, these algorithms can take advantage of spatial distribution of transmitters to allow more interference free transmissions in each time slot.
than tree-based algorithms.

Hung et al. [69] presented a greedy heuristic algorithm for the MLBS problem under the RTS/CTS interference model. In each time slot, they first select a covering set from nodes with the broadcast message to cover all those that have yet to receive said message. Basically, they recursively choose into the covering set a node that covers the most nodes that have yet to receive the message. Then, they construct an independent-transmission set, which nodes share common neighbors, meaning they can transmit simultaneously. In each time slot, a node in the covering set with more uncovered neighbors will have a higher priority to be selected into the independent-transmission set, but its neighbors will be removed from this set.

Hung et al. [69] also proposed a distributed algorithm based on their centralized method. In their distributed implementation, each informed node $v$ is required to construct a covering set and a series of independent-transmission sets based on its two hops neighbors’ information. Briefly, in node $v$’s view, the covering set only contains $v$’s one-hop uninformed neighbors and needs to cover all of node $v$’s two-hop uninformed neighbors. Then, a series of independent-transmission sets of node $v$ is iteratively constructed until all nodes in the covering set are considered. Nodes in each independent-transmission set can broadcast at the same time. Next, node $v$ assigns its one-hop uninformed neighbors with different transmission sequences if they are not in the same independent-transmission set. However, their distributed algorithm cannot be guaranteed to be collision-free and does not give any guarantees with respect to the number of transmissions and broadcast latency.

Jiang [45] also proposed a greedy heuristic algorithm for the MLBS problem. To schedule more interference-free transmissions from multiple layers of a broadcast tree, in each time slot, their algorithm adopts a greedy coloring method to label informed nodes, where nodes with the same color can transmit without interference. Specifically, in their method, nodes with more uninformed neighbors will be colored first. Then, they require nodes with the first color transmit immediately. Different from conventional methods where nodes with other colors transmit subsequently [14] [55], their heuristic algorithm will recolor nodes of other colors with those receivers of nodes with the first color. Therefore, more interference-free trans-
missions are selected in the new coloring set.

Mahjourian et al. [55] and Tiwari et al. [75] study a series of greedy heuristic algorithms for the MLBS problem under the protocol interference-model. To select the maximal number of interference-free transmissions in each time slot, they study a variety of heuristic criteria to give higher priority to particular transmissions in a set of interfering transmissions, such as the number of neighbors that have yet to receive the broadcast. That is, transmissions with a higher priority will be scheduled first. In their study, they compare heuristic algorithms that adopt different criteria with the optimal algorithm. The average approximation ratio is defined as the ratio between the broadcast latency of heuristic algorithms and that of BFS. As Table 2.3 shows, when the heuristic criterion is the number of neighbors that have yet to receive the broadcast, the corresponding greedy heuristic algorithm in [75] produces a near optimal performance with respect to broadcast latency, with an average approximation ratio of 1.52. The main reason is because this criterion, i.e., the number of neighbors that have yet to receive the broadcast, allows maximal number of nodes to be informed with each new transmission.

### Table 2.3
Comparison of various heuristics algorithms [75]

<table>
<thead>
<tr>
<th>Heuristic criteria</th>
<th>Average approximation ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Larger number of neighbors that have yet to receive the broadcast</td>
<td>1.52</td>
</tr>
<tr>
<td>Larger number of BFS descendents</td>
<td>2.32</td>
</tr>
<tr>
<td>Larger Euclidean hops to the source</td>
<td>2.43</td>
</tr>
<tr>
<td>Larger node ID (Random)</td>
<td>2.59</td>
</tr>
<tr>
<td>Larger number of neighbors</td>
<td>2.79</td>
</tr>
<tr>
<td>Larger distance to the source</td>
<td>2.83</td>
</tr>
<tr>
<td>Smaller Euclidean hops to the source</td>
<td>3.69</td>
</tr>
<tr>
<td>Smaller distance to the source</td>
<td>6.02</td>
</tr>
</tbody>
</table>
2.3 MLBS over Duty-cycled WSNs

To date, only a handful of works have proposed broadcast algorithms over duty-cycled WSNs. Existing broadcast algorithms can be categorized into unicast, cluster and duty cycle aware.

2.3.1 Unicast Based Methods

In this category, broadcast is carried out via unicast communication. That is, a node transmits a broadcast message multiple times to its neighbors. The advantage of this method is that it guarantees every node receives the broadcast message. The downside, however, is that it cost more energy and result in longer delays because each sender, and intermediate nodes, needs to deliver a message multiple times.

Sun et al. [72] proposed an Asynchronous Duty-cycle Broadcasting (ADB) algorithm where a node uses unicast to transmit a broadcast message to its neighbors. ADB relies on RI-MAC [73], which is a receiver initiated MAC protocol. Briefly, in RI-MAC, a node that intends to send a message to its neighbor will stay awake to wait for a beacon from the intended receiver. When the intended receiver wakes up, it will examine whether the channel is idle. If it is, it immediately transmits a beacon message, indicating that it is awake and ready to receive a message. Upon reception of a beacon, the sender starts its transmission immediately. In ADB, the authors define two broadcast status for each node: reached and delegated. A reached status means a node has received the broadcast message, and delegated status denotes a node with the broadcast message that is refrained from rebroadcasting. Each node adopts a footer, which is added to all data and acknowledgement messages to determine a node’s status. Moreover, the footer can also be used to indicate link quality; using either four bits link quality estimation [25] or short-term wireless link quality estimation [7]. The link quality information can then be used to decide a node’s delegated status. For instance, consider a network with three nodes: $A$, $B$ and $C$. All nodes are within each other’s transmission range. If node $B$ wants to transmit a message to $C$, and node $A$ finds it has a better link quality to node $C$ as compared to node $B$ to $C$, node $A$ will then delegate node $B$ to transmit the message to node $C$. 
That is, node $B$ marks itself as delegated and goes back to sleep.

Lee et al. [50] described a pseudo-random asynchronous duty-cycle MAC protocol, which is based on RI-MAC. The protocol uses a hash function to determine a node’s wake-up schedule. This hash function takes node’s ID as input. This means if a node knows its neighbors’ ID, it can also learn its neighbor’s wake-up schedule through this hash function. Thus, if a node wants to deliver a message to its neighbor, it will wake up at the computed time rather than waiting idly for the receiver to wake-up.

### 2.3.2 Cluster Based Methods

Cluster-based methods group nodes with similar wake-up time together and transmit a broadcast message to these nodes simultaneously. The advantage, in comparison to unicast-based methods, is that it consumes less energy and time to send a message to its neighbors. The algorithm by Hurni et al. [42] is based on Wise-MAC [21], which is an energy efficient MAC protocol that uses a preamble sampling technique (or low power listening). Generally, if a sender tries to deliver a message to its neighbor, it needs to send a preamble message, which is long enough to guarantee the intended receiver, is woken up. When the intended receiver wakes up and detects the preamble, it will stay active to receive data. The main idea introduced by Wise-MAC is letting each node learn the wake-up schedule of all its neighbors through HELLO messages. Consequently, the duration of a preamble message is shortened and the energy consumption is reduced. With the knowledge of neighbors’ wake-up time, Hurni et al.’s algorithm requires a sender to group neighbors with near wake-up intervals and transmits a broadcast message to neighbors, which are in the same group simultaneously. Wake-up intervals are considered to be near if the difference between their closest wake-up times is smaller than the time it takes to transmit the preamble and the data message.

Lai et al. [49] presented a hybrid broadcast protocol for the MLBSDC problem. In order to receive beacon messages from all neighbors in a collision-free manner, they apply a quorum system to organize the wake-up slots for each node, where a quorum system is a subset of time slots, where every two of which intersect, and each subset is called a quorum. This means two neighboring nodes that adopt such a quorum
system as their wake-up schedule are able to communicate with each other without collision. When a sender receives a beacon message from the first awake neighbor, it will remain active for several time slots, called waiting time, in order to wait for more nodes to wake up, rather than transmitting the broadcast message immediately. When the waiting time expires, the sender will send the broadcast message immediately and awake neighbors will remain active and receive the message. By doing this, more neighbors that wake up before the waiting time expires are able to receive the broadcast message. As a result, a transmitter avoids redundant transmissions by increasing the group of neighbors that receive messages at the same time. To further reduce redundant transmissions, their algorithm selects a small number of relaying nodes among one-hop nodes that are awake to cover two-hop neighbors, which is similar to the MPR method [66].

2.3.3 Duty Cycle Aware Methods

Duty cycle aware broadcast protocols exploit the fact that each node is aware of their neighbors’ wake-up schedule, which can be obtained through periodic beacon messages. If a node wants to transmit a message, it will wake up at the corresponding receiver’s wake-up time slot to send the message. As a result, nodes incur minimal idle listening time and avoid long preamble communication, which consumes a non-negligible amount of energy [73].

Guo et al. [35] proposed an opportunistic flooding algorithm for duty-cycled WSNs over unreliable links. Initially, they construct an energy-optimal tree as follows. At each layer $i$ of the BFS, it assigns nodes in layer $i$ with a parent chosen from nodes in layer $i - 1$ that have the best link quality. To reduce latency, their flooding algorithm allows the use of links outside the energy-optimal tree if transmissions through these links allow receivers to receive a message quicker. To do this, they require each node to compute a one-hop delay $D_p$ based on active time slots and the link quality between its parent and itself. The variable $D_p$ indicates the expected latency for a node to receive a message successfully from its parent. After receiving a new broadcast message, node $v$ will calculate the one-hop delay $D_n$ for its one-hop neighbor $u$. Here, $D_n$ is the expected latency for node $u$ to receive a message successfully
from node $v$. Then, node $v$ compares $D_n$ with $u$’s $D_p$. If $D_n \leq D_p$, it concludes that the message sent by node $v$ can reach node $u$ earlier than $u$’s parent. As a result, this transmission from $v$ to $u$ is needed. If $D_n > D_p$, node $u$ first gets the message from its parent. Hence, the transmission from $v$ to $u$ is considered redundant. To reduce the chance of collisions, they require a sender to back off for a period of time according to the link quality between the sender and its intended receiver.

Wang et al. [83] outlined a broadcast algorithm based on constructing a time coverage graph to determine broadcast times. This time-coverage graph is presented as a grid. A vertex in the grid represents the set of the nodes that have received a broadcast message, denoted by $RE$. There are two kinds of edges in the grid: time and forward. A time edge connects two adjacent vertices in the same row, which indicates time $t$ is variable but the set $RE$ remains fixed. In contrast, an edge of type forward connects two neighboring vertices in different rows, which indicates a broadcast transmission. In other words, one or more active nodes in set $RE$ transmit a message to their neighbors, which leads to a new set $RE$ with a larger size. Therefore, the size of set $RE$ increases from the first row to the last row, and the set $RE$ of the last row contains all nodes. Moreover, they assign each edge with a weight $w$ which is a linear combination of the wake-up interval between senders and their receivers, and the number of senders in $RE$ forwarding the broadcast message at time $t$. After the time-coverage graph is constructed, the broadcast problem is transformed into finding the shortest path problem from the first row to the vertices in the last row in the weighted grid.

In [83], Wang et al. also described a distributed algorithm. Each node maintains a covering set of one-hop and two-hop neighbors that have received the broadcast message. A node constructs a time-coverage graph based on the covering set. For instance, the first row’s $RE$ is the node itself and the last row’s contains the sender and its one-hop, two-hop neighbors. Whenever a node $v$ receives a message, it will update its covering set first based on its two hops information. Then, starting from the row that contains $v$’s covering set in the time-coverage graph, node $v$ first calculates the broadcast time for its one-hop neighbors. Then, $v$ embeds the calculated time in the broadcast message before broadcasting to its one-hop neighbors. Moreover,
to address the problem of inconsistent broadcast times, where a node may receive different broadcast times for the same node from its neighbors, the node will overhear messages from its neighbors and examine whether the same node in its covering set follows the same broadcast time. If not, the node will re-compute the broadcast time with an updated covering set. However, both centralized and distributed algorithms depicted in [83] do not take into consideration interference, which cause packet loss. Moreover, the construction of a time-coverage graph for a whole network is an NP-hard problem.

Hong et al. [38] propose two approximation algorithms for the MLBSDC problem under the RTS/CTS interference model. The algorithms start by constructing a shortest path tree (SPT) rooted at the source node. The cost of each edge is defined as the one-hop broadcast latency that corresponds to the wake-up interval between two neighboring node. They divide the SPT into different layers, where each layer contains the set of nodes with the same minimal cost to the source. Next, they schedule the transmissions layer by layer based on the SPT in a top-down manner. To be specific, in their first algorithm, for each layer $i$ of the SPT, nodes sharing the same neighbors will be assigned with a different transmission time to avoid collision. They prove for each layer $i$, the first algorithm takes at most $(\Delta^2 + 1)T$ time slots to inform all nodes in layer $i$, where $T$ is a duty-cycled scheduling period. To further reduce the broadcast latency produced by the first algorithm, they proposed an enhanced algorithm, whereby it first constructs a MIS from the SPT in a top-down manner. Then, for each layer, a dominator is selected as the parent of nodes not in the MIS. A node not belonging to MIS is only selected as the parent of dominators. Next, they color dominators in each layer with a different color based on the rule that dominators that have a common neighbor will be labelled with a different color. During scheduling, a dominator will be assigned a transmission time based on its own color. However, the transmission of the parent of dominators is scheduled based on its children’s colors. They show their enhanced algorithm gives a $24T$-approximation solution with respect to broadcast latency.

Jiao et al. [46] also outlined a constant approximation algorithm called OTAB for MLBSDC problem under the RTS/CTS interference model with a ratio of $17T$ for
broadcast latency. OTAB starts by constructing a broadcast tree following a two-step method. In the first step, they divide nodes into different subsets according to their active time slots, and then for each subset, they construct a MIS induced by SPT rooted at the source in an increasing order. In the second step, for each layer \( i \) of the SPT, the parent of dominators in layer \( i \) are chosen from higher layers, and dominators in layer \( i \) are selected as the parent of nodes in layer \( i \) or layers lower than \( i \) with the same active time slot. The choosing process is based on the rule that if a node covers most nodes that do not have a parent yet, this node is first chosen as the parent node. To avoid collision, OTAB then labels the dominators in each MIS and their parent nodes with a different color, respectively, if they have common neighbors. Next, OTAB schedules the transmissions from parents to their children based on parents’ color layer by layer in the top-down manner. They prove OTAB gives a \( 17T \)-approximation solution for broadcast latency, and a 15-approximation solution for number of transmissions.

Xu et al. [88] extend the pipelined algorithm in [41] to duty-cycled scenario and prove it produces an approximation algorithm with a ratio of \( RT + O(\log_2 R)T \) for broadcast latency. Xu et al.’s algorithm starts by constructing a ranking tree in the same way as [41]. Different from [41], for each sender in a pipelined session, defined as transmissions from parents in layer \( i \) with rank \( r \) to their children in layer \( i + 1 \), it transmits in the \( i + 51(r_0 - r) + 3k \)-th period when its children are awake, where \( k \) is the color assigned to the sender and \( r_0 \) is the rank value of the source.

### 2.4 Summary

As listed in Table 2.4, most current algorithms for MLBS problem adopt tree-based method to schedule transmissions. Briefly, these algorithms rely on a broadcast tree constructed using BFS, which in turn is used to schedule interfering transmissions. However, except for [26], [27] and the pipelined algorithm in [41], other tree-based algorithms utilize a layer-by-layer approach to schedule interfering transmissions, where all nodes in a BFS layer must be informed before the broadcast proceeds to subsequent layers. The intention behind this scheme is to avoid potential interference
### Table 2.4
Comparison of MLBS algorithms

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Types</th>
<th>Groups</th>
<th>Interference models</th>
<th>Networks</th>
<th>Approximation ratio&lt;sup&gt;a&lt;/sup&gt;</th>
<th>Approximation ratio&lt;sup&gt;b&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chanpitac et al. [16]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Always-on</td>
<td>(O(\Delta))</td>
<td>-</td>
</tr>
<tr>
<td>Gandhi et al. [27]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Always-on</td>
<td>(O(1))</td>
<td>-</td>
</tr>
<tr>
<td>Huang et al. [41]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Always-on</td>
<td>24</td>
<td>8</td>
</tr>
<tr>
<td>Huang et al. [41]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Always-on</td>
<td>16</td>
<td>8</td>
</tr>
<tr>
<td>Gandhi et al. [26]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Always-on</td>
<td>(1 + O(\frac{\log R}{R}))</td>
<td>-</td>
</tr>
<tr>
<td>Hung et al. [69]</td>
<td>Centralised/Distributed</td>
<td>Greedy heuristic</td>
<td>RTS/CTS</td>
<td>Always-on</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Chen et al. [14]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>Protocol</td>
<td>Always-on</td>
<td>(O((\frac{r_T}{r_T})^2))</td>
<td>-</td>
</tr>
<tr>
<td>Huang et al. [40]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>Protocol</td>
<td>Always-on</td>
<td>(6 \left(\frac{2}{3} \left(\frac{r_T}{r_T} + 2\right)^2\right))</td>
<td>-</td>
</tr>
<tr>
<td>Tiwari et al. [75]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>Protocol</td>
<td>Always-on</td>
<td>(2 \left(\frac{2}{3} (\frac{r_T}{r_T} + 1)^2 \chi^2 + \frac{2}{3} \left(\frac{r_T}{r_T} + 1\right) \chi + \frac{2}{3}\right)^2)</td>
<td>-</td>
</tr>
<tr>
<td>Tiwari et al. [75]</td>
<td>Distributed</td>
<td>Greedy heuristic</td>
<td>Protocol</td>
<td>Always-on</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Tiwari et al. [75]</td>
<td>Centralised</td>
<td>Greedy heuristic</td>
<td>Protocol</td>
<td>Always-on</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Mahjourian et al. [55]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>Protocol</td>
<td>Always-on</td>
<td>(O(\max(\frac{r_T}{r_T}, \frac{r_T}{r_T})^2))</td>
<td>-</td>
</tr>
<tr>
<td>Mahjourian et al. [55]</td>
<td>Centralised</td>
<td>Greedy heuristic</td>
<td>Protocol</td>
<td>Always-on</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Jiang et al. [45]</td>
<td>Centralised</td>
<td>Greedy heuristic</td>
<td>Protocol</td>
<td>Always-on</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Huang et al. [40]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>Physical</td>
<td>Always-on</td>
<td>(O((\frac{r_T}{r_T})^2))</td>
<td>-</td>
</tr>
<tr>
<td>Wan et al. [81]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>Physical</td>
<td>Always-on</td>
<td>(O(1))</td>
<td>-</td>
</tr>
<tr>
<td>Wan et al. [81]</td>
<td>Centralised/Distributed</td>
<td>Greedy heuristic</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Hong et al. [38]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>(O((\Delta^2 + 1)T))</td>
<td>-</td>
</tr>
<tr>
<td>Hong et al. [38]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>(24T)</td>
<td>-</td>
</tr>
<tr>
<td>Jiao et al. [46]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>(17T)</td>
<td>15</td>
</tr>
<tr>
<td>Xu et al. [88]</td>
<td>Centralised</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>(T + O(\frac{2\sigma^2 R}{R})T)</td>
<td>-</td>
</tr>
<tr>
<td>Guo et al. [35]</td>
<td>Distributed</td>
<td>Tree-based</td>
<td>RTS/CTS</td>
<td>Duty-cycled</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

<sup>a</sup>This approximation ratio is with regards to broadcast latency  
<sup>b</sup>This approximation ratio is with respect to number of transmissions
from subsequent layers. However, the key problem is interference-free transmissions in subsequent layers will be blocked until all nodes in upper layers have finished their reception. In contrast, the tree-based approaches proposed in Chapter 3 and 5, namely BS-1, BS-2, IABBS and IAEBS, aim to assign nodes with transmission times based on a greedy mechanism. Briefly, in order for a transmission to be interference-free, there must be no nodes among the interference range of a sender (respectively, its intended receiver) receiving (respectively, transmitting) at the same time. Consequently, they are able to schedule simultaneous transmissions in multiple layers of a BFS tree, which helps to reduce broadcast latency. In particular, BS-2 produces the best constant approximation ratio of $13T$ as compared to the state of the art algorithm reported in [46], which is $17T$.

To avoid the layer-by-layer approach adopted by tree-based algorithms, another approach is to use a greedy heuristic method. This involves considering all nodes with the broadcast message as potential transmitters. However, only transmitters/nodes with a higher priority as per some heuristic criteria are allowed to transmit. Therefore, greedy heuristic algorithms are able to schedule transmissions across multiple layers concurrently, which has the effect of speeding up the propagation of a broadcast message. As shown by the extensive simulation results of [45] [55] [69] and [75], on average, greedy heuristic algorithms have a near optimal performance in terms of broadcast latency. However, none of them are able to guarantees any bounds on broadcast latency. As we will see in Chapter 4 and 6, the proposed algorithms reported, namely CEN/DIS and HBA, are able to give a constant approximation ratio for broadcast latency, and achieve a near optimal performance in terms of broadcast latency on average case.

For the MLBS problem under the protocol interference model, all the works reviewed thus far, except [14] and [45], apply the same geometrical constraint to avoid interfering transmissions. For example in [40], [55] and [75], senders or receivers with a distance less than $(r_I + r_T)$ must not be scheduled to transmit or receive at the same time. However, this geometrical constraint is in general stronger than what is needed to avoid interfering transmissions. That is, it is possible for two parallel transmissions to receive a message correctly despite not satisfying this geometrical constraint. In
the ensuing chapters, Chapter 5 addresses the aforementioned limitations by proposing two constant approximation algorithms, IABBS and IAEBS. Unlike past works, IABBS and IAEBS do not use such geometrical constraint. Instead, these algorithms allow parallel transmissions to proceed only if they do not interfere with each other, which leads to lower broadcast latencies.

Further, as shown in Table 2.4, existing algorithms for MLBSDC problem have only considered the RTS/CTS model. To bridge this gap, in Chapter 5 and 6, this thesis proposes IABBS and IAEBS to address MLBSDC problem under the protocol and physical interference models. To date, these algorithms are the first attempt to address the MLBSDC problem in these scenarios. Furthermore, from Table 2.4, we can see only few works have outlined distributed algorithms for the MLBS problem. To this end, Chapter 4 and 6 proposed two distributed algorithms, called CEN/DIS and HBA, which only require a node’s two-hop neighbors information. Unlike the distributed algorithms reported in [35] [69] and [81], where they do not provide any guarantees on interference-free transmission no broadcast latency, CEN/DIS and HBA ensure all transmissions are interference-free and produce a constant upper-bounded for broadcast latency.
Chapter 3

Approximation Algorithms under the RTS/CTS Interference Model

As mentioned in Chapter 1, given a source node with a packet to broadcast, the aim is to propagate the packet to all nodes in a collision free manner whilst incurring minimum latency. This problem, called Minimum Latency Broadcast Scheduling (MLBS), has been studied extensively in wireless ad-hoc networks assuming nodes are awake at times, and has been shown to be NP-hard [27]. However, only a few studies have addressed this problem in the context of duty-cycled WSNs. The key difference is that nodes do not wake-up simultaneously, and hence, not all neighbors of a transmitting node will receive a broadcast packet at the same time. Unfortunately, the problem remains NP-hard [38] and multiple transmissions may be necessary due to different wake-up times.

Henceforth, this chapter considers MLBS in Duty-Cycled WSNs (or MLBSDC) under the RTS/CTS interference model and presents two approximation algorithms, BS-1 and BS-2, that produce a maximum latency of at most \((\Delta - 1)TH\) and \(13TH\) respectively. Here, \(\Delta\) is the maximum degree of nodes, \(T\) denotes the number of time slots in a scheduling period, and \(H\) is the broadcast latency lower bound obtained from the shortest path algorithm. BS-1 and BS-2 are evaluated under different network configurations via simulation. The results confirmed that the latencies achieved by them are much lower than existing schemes. In particular, compared to OTAB [46], the best broadcast scheduling algorithm to date, the broadcast latency
and number of transmissions achieved by BS-1 is at least $\frac{1}{17}$ and $\frac{2}{5}$ of that of OTAB respectively.

## 3.1 Preliminaries

### 3.1.1 Network Model

It is assumed that all nodes in a WSN have an equal transmission range $r_T$. Hence, the network can be modelled as a Unit Disc Graph (UDG), $G = (V, E)$, where $V$ is the set nodes, and $E$ represents the set of edges/links that exist between two nodes if their Euclidean distance is no more than $r_T$. In this chapter, the network adopts the RTS/CTS interference model, where a message is considered lost if there is a collision; i.e., two or more simultaneous transmissions to a common node. Moreover, a node must not receive and send a message at the same time. Let $N_1(v)$ denote the set of one-hop neighbors of node $v \in V$ and $n = |V|$ is the number of nodes. Accordingly, $N_2(v)$ denotes the two-hop neighbors of $v$.

Assume that the scheduling period is divided into $T$ unit slots with fixed and equal length, denoted by $0, 1, 2, 3, \cdots, T - 1$. Here, a message can be successfully delivered from a sender to a receiver within a time slot [33]. The network is locally synchronized at a slot level. As shown in [33], this can be achieved using local synchronization techniques, such as the flooding time synchronization protocol (FTSP) in [56], which requires each node keeps the clock drift information of its neighbors. FTSP can yield an accuracy of 2.24 $\mu$s, achieved at a low cost amounting to a few bytes of packet exchanges among neighboring nodes every 15 minutes. It’s important to note that this accuracy is sufficient as the active duration of each node is typically above 10,000 $\mu$s [20] [32]. Moreover, transmissions are not required to start at the beginning of each slot, meaning nodes do not need tight synchronization in order to communicate. Without loss of generality, assume the clock drift between any two nodes is zero.

Given the scheduling period $T$, the duty cycle is thus defined as the ratio between active time and $T$. For example, if $T = 10$, a 10% duty cycle means nodes are only
awake in one slot. Similar to duty cycle aware works listed in Section 2.3.3, the pseudo-random asynchronous MAC model, which is duty-cycle aware, such as [50] and [74], is considered in this chapter, where each node uses a pseudo-random sequence to generate its own active/sleep schedule independently, and hence, a sender, who knows its intended receiver’s seed and last active time, can easily learn the receiver’s next active time. Specifically, each node $v$ selects one active time slot $\tau(v)$ from $[0, \ldots, T - 1]$ randomly and independently, and wakes up at this time slot to receive a message. If node $v$ wants to transmit a message, it can wake-up at any time slot as long as the receiver is awake and there is no collision.

### 3.1.2 Graph Definitions and Theories

An Independent Set (IS) $D$ in $G(V, E)$ is defined as a subset of $V$ such that $\forall u, v \in D, (u, v) \notin E$. A maximal independent set (MIS) $U$ is an independent set which is not a subset of any other independent sets. MIS is also called independent dominating set which means every vertex not in MIS is joined to at least one member of MIS by some edge for a graph $G$, hence, vertices in MIS are also called dominators. It is known that for a UDG, a node can be adjacent to at most five nodes in an IS set [24] and in a circle of radius two, the number of points whose mutual distance is at least one does not exceed 19 with one point at the centre. That is, the size of the IS is no more than 19 [6]. Note that, a MIS $U$ is a dominating set of $G$, since any node $v \in V \setminus U$ will be adjacent to some nodes in $U$. The authors in [80] show that the size of $U$ for a UDG graph is bounded by $O(R^2)$, where $R$ is the maximum hop distance from the source node $s$ to other nodes. In addition, $U$’s size does not exceed $4\text{opt} + 1$, where $\text{opt}$ denotes the minimum size of a Connected Dominating Set (CDS) [79] where CDS is defined as a dominating set $C$ of a graph $G$ and the subgraph induced by $C$ is connected. A summary of notations used in this chapter can be found in Table 3.1.

### 3.1.3 Problem Formulation

This chapter studies the one-to-all MLBSDC problem, whereby a source node $s \in V$ emits a message which is then broadcast to all other nodes. The broadcast finishes
Table 3.1
Definition of Notations in Chapter 3

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(V, E)$</td>
<td>Network graph</td>
</tr>
<tr>
<td>$U_i$</td>
<td>The MIS of graph $G$</td>
</tr>
<tr>
<td>$N_1(v)$</td>
<td>Node $v$’s one-hop neighbors</td>
</tr>
<tr>
<td>$U$</td>
<td>Nodes in $U \cap L_i$</td>
</tr>
<tr>
<td>$T$</td>
<td>Scheduling period</td>
</tr>
<tr>
<td>$M$</td>
<td>Nodes in $V \setminus U$</td>
</tr>
<tr>
<td>$H$</td>
<td>Broadcast latency lower bound</td>
</tr>
<tr>
<td>$M_i$</td>
<td>Nodes in $M \cap L_i$</td>
</tr>
<tr>
<td>$\tau(v)$</td>
<td>Node $v$’s active slot</td>
</tr>
<tr>
<td>$Lat(u, v)$</td>
<td>Cost of edge $(u, v)$</td>
</tr>
<tr>
<td>$\tau_i$</td>
<td>Layer $i$’s active slot</td>
</tr>
<tr>
<td>$C(v, i)$</td>
<td>Node $v$’s children in layer $i$</td>
</tr>
<tr>
<td>$T_{SPT}$</td>
<td>Shortest path tree (SPT)</td>
</tr>
<tr>
<td>$P(v)$</td>
<td>Node $v$’s parent node</td>
</tr>
<tr>
<td>$T_b$</td>
<td>Broadcast tree</td>
</tr>
<tr>
<td>$L_i$</td>
<td>Nodes in layer $i$ of $T_{SPT}$</td>
</tr>
<tr>
<td>$l$</td>
<td>Maximum number of layers in $T_{SPT}$</td>
</tr>
<tr>
<td>$rec(v)$</td>
<td>Reception time of $v$</td>
</tr>
<tr>
<td>$I_1(v)$</td>
<td>Reception time of nodes in $N_1(v) \setminus C(v, i)$ receiving a message</td>
</tr>
<tr>
<td>$tr1(v, i)$</td>
<td>Transmission time of $v$ sending a message to its children of layer $i$ in Phase 1 of BS-2</td>
</tr>
<tr>
<td>$I_2(v)$</td>
<td>Reception time of one-hop neighbors of nodes in $C(v, i)$ receiving a message</td>
</tr>
<tr>
<td>$tr2(v, i)$</td>
<td>Transmission time of $v$ sending a message to its children of layer $i$ in Phase 2 of BS-2</td>
</tr>
<tr>
<td>$I(v)$</td>
<td>$I_1(v) \cup I_2(v)$</td>
</tr>
<tr>
<td>$Y$</td>
<td>Nodes in $V \setminus X$</td>
</tr>
<tr>
<td>$tr(v, i)$</td>
<td>Transmission time of $v$ sending a message to its children of layer $i$ in BS-1</td>
</tr>
<tr>
<td>$X$</td>
<td>Nodes in $U$ and non-leaf nodes of $M$ in $T_b$</td>
</tr>
<tr>
<td>$r_i$</td>
<td>Maximum reception time of nodes in layer 0 to $i$</td>
</tr>
<tr>
<td>$R$</td>
<td>Maximum hop distance from the source to any node</td>
</tr>
</tbody>
</table>
when all nodes receive the message successfully. Without loss of generality, define the start time of node s’s broadcast to be $t_0$, and the broadcast latency is the maximum time taken by a message to reach all nodes in the network.

To minimize broadcast latency, the MLBSDC problem is modelled as follows. Let $(S_i, t_i)$ denote the $i^{th}$ transmission, where the set of nodes in $S_i$ transmit a broadcast message at time $t_i$, where $i \in \mathbb{Z}$, $t_i$ is the active time of nodes that receive the message from nodes in $S_i$ at the $i^{th}$ transmission, and $C(S_i)$ denotes the set of nodes that received the message from nodes in $S_i$ interference-free. Given a wireless network $G(V, E)$ with duty cycle, and a source node $s \in V$, the MLBSDC problem is to find a forwarding schedule,

$$S = \{(s, t_0), (S_1, t_1), \ldots, (S_m, t_m)\}$$

(3.1)

that satisfies the following constraints: (i) $t_0 < t_1 < \cdots < t_m$, (ii) any node in $S_i$ cannot be scheduled to transmit the message until it receives the message, (iii) all transmissions from $S_i$ to $C(S_i)$ must be interference-free, (iv) $|\bigcup_{i=1}^{m} C(S_i)| = |V|$, and broadcast latency of $(t_m - t_0)$ is minimum. In other words, to address MLBS problem needs to find an interference-free broadcast schedule that guarantees all nodes in $V$ receive the broadcast message interference-free in minimum time.

### 3.2 Proposed Algorithms

In this section, two approximation algorithms are outlined for the MLBSDC problem: BS-1 and BS-2. BS-1 is an algorithm with a broadcast latency of at most $(\Delta - 1)TH$. BS-2 is an enhancement of BS-1, producing a broadcast schedule with latency of at most $13TH$, and has a better performance than BS-1 in dense networks, because its broadcast latency is bounded by a constant number 13, and thus is independent of $\Delta$ and number of nodes.

#### 3.2.1 BS-1

The main idea of BS-1 is to schedule transmissions in a greedy manner, and also allow a node in a lower layer to transmit or receive earlier than a node in an upper
layer. In the following explanation, Figure 3.1 and 3.2 will be used to illustrate key aspects of BS-1. The network in Figure 3.1 consists of 12 nodes, and node $s$ is the source node. Set $T = 4$, i.e., $[0, 1, 2, 3]$, and the active time slots of all nodes are listed in Table 3.2.

![Figure 3.1: The topology of network $G$](image)

### Table 3.2

<table>
<thead>
<tr>
<th>ID</th>
<th>$s$</th>
<th>$v_1$</th>
<th>$v_2$</th>
<th>$v_3$</th>
<th>$v_4$</th>
<th>$v_5$</th>
<th>$v_6$</th>
<th>$v_7$</th>
<th>$v_8$</th>
<th>$v_9$</th>
<th>$v_{10}$</th>
<th>$v_{11}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Layer</td>
<td>$L_0$</td>
<td>$L_1$</td>
<td>$L_1$</td>
<td>$L_2$</td>
<td>$L_2$</td>
<td>$L_2$</td>
<td>$L_3$</td>
<td>$L_3$</td>
<td>$L_4$</td>
<td>$L_5$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

BS-1 takes as input $G(V, E)$ and a source node $s \in V$. BS-1 firstly constructs a shortest path tree (SPT) $T_{SPT}$ rooted at node $s$. $T_{SPT}$ can be obtained by applying Dijkstra’s algorithm [18] on $G(V, E)$ and setting the cost of each edge $(u, v) \in V$ to $Lat(u, v)$. Here, $Lat(u, v)$ is defined as follows [46],

$$
Lat(u, v) = \begin{cases} 
\tau(v) + 1, & \text{if } u = s; \\
\tau(v) - \tau(u), & \text{if } u \neq s \text{ and } \tau(v) - \tau(u) > 0; \\
\tau(v) - \tau(u) + T, & \text{otherwise}
\end{cases}
$$

In other words, the latency incurred on each edge is the duration defined as the time slot in which a sender $u$ has a message ready for transmission to the time a receiver $v$ becomes active or wakes up; i.e., ready to receive a message from $u$. Gu et al. [33] showed that sleep latency is usually in the order of seconds, which is orders of magnitude longer than other delivery latencies such as processing, queuing and
Figure 3.2: An illustration of BS-1. The label (rec, tr) denotes the reception and transmission time of a node.

Once $T_{SPT}$ is constructed, divide nodes into different layers according to their cost to source node $s$. This also means nodes at the same layer share the same active time slot; i.e., all nodes that belong to the same layer wake up simultaneously. Let $L_i$, where $i = 0, 1, 2, \cdots, l$, be the set of nodes at layer $i$ and $l$ is the maximum layer number of $T_{SPT}$. For example, in Figure 3.2, $l = 5$, $L_1 = \{v_1, v_2\}$ and both nodes in $L_1$ share the same active time slot 0, i.e., $\tau(v_1) = \tau(v_2) = 0$. Denote by $H$ the cost from nodes in layer $l$ to source node $s$; That is, $H$ is the broadcast latency lower bound of the network, i.e., $H = 6$ in Figure 3.2. In addition, it is clear that $H \geq l$.

The next step is to construct a broadcast tree $T_b$. Unfortunately, due to the collisions that exist in $T_{SPT}$, the broadcasts cannot be scheduled directly based on it. Instead, BS-1 starts by constructing a broadcast tree $T_b$ based on $T_{SPT}$, and then schedules the broadcast according to the broadcast tree. Let $P(v)$ be the parent of node $v$ in $T_b$ and $C(u, i)$ be the set of node $u$’s children which lie in layer $i$.
requires to determine nodes’ parent-children relationship layer by layer in a top-down manner. To efficiently reduce transmission times, Algorithm 3.1 selects a parent node that covers the most nodes in a given layer. Specifically, at each layer $i$, where $0 < i \leq l$, a parent node $u$ is preferentially selected from an upper layer $j$, where $j < i$, and node $u$ must cover the most nodes in $L_i$ that have yet to be assigned a parent (line 7 in Algorithm 3.1). For instance in Figure 3.2, for nodes in $L_2 = \{v_3, v_4, v_5, v_6\}$, initially, node $v_1$ from layer 1 covers two nodes of layer 2, i.e., $v_3$ and $v_6$, node $v_2$ covers three nodes of layer 2, i.e., $v_4$, $v_5$ and $v_6$, and node $s$ only covers node $v_6$ in layer 2. Hence, node $v_2$ is first selected as the parent node of node $v_4$, $v_5$ and $v_6$, i.e., $P(v_4) = P(v_5) = P(v_6) = v_2$ and $C(v_2, 2) = \{v_4, v_5, v_6\}$, as $v_2$ covers most nodes in layer 2. Similarly, node $v_1$ is then chosen as the parent node of node $v_3$. The other layers are handled in a similar manner, and the final result of $T_b$ is shown in Figure 3.2. Given that a parent node has the highest coverage, it is thus responsible for delivering the broadcast message to all of its children.

**Algorithm 3.1 Broadcast Tree $T_b$ Construction for BS-1**

1: $T_b \leftarrow (V_b, E_b), V_b \leftarrow V, E_b \leftarrow \emptyset$
2: $T_{SPT} \leftarrow$ shortest path tree in $G$ rooted at $s$
3: $P(v) \leftarrow \text{NIL}, \forall v \in V$
4: for $i \leftarrow 1$ to $l$ do
5: $L_i' \leftarrow L_i$
6: while $L_i' \neq \emptyset$ do
7: $u \leftarrow$ a node in $L_{j<i}$ with maximum
   $\left| \{v \mid v \in N_1(u) \cap L_i \text{ and } P(v) = \text{NIL} \} \right|$ value
8: $C(u, i) \leftarrow \{v \mid v \in N_1(u) \cap L_i \text{ and } P(v) = \text{NIL} \}$
9: $P(v) \leftarrow u$ and $L_i' \leftarrow L_i' \setminus \{v\}, \forall v \in C(u, i)$
10: end while
11: end for
12: $E_b \leftarrow \{(u, v) \mid u = P(v)\}$
13: return $T_b = (V_b, E_b)$

After constructing broadcast tree $T_b$, the transmissions from the parent nodes are scheduled based on $T_b$ as per Algorithm 3.3. This scheduling starts at time slot 0 and works layer by layer in the top-down manner. Let $tr(v, i)$ and $rec(v)$ denote
node $v$’s scheduled transmission time and reception time respectively, where $tr(v, i)$
denotes the transmission time at which node $v$ sends a message to nodes in layer
$i$, e.g., $tr(s, 1) = 0$, and $rec(v_1) = rec(v_2) = 0$; see Figure 3.2. Note that, a
node may be assigned multiple transmission time slots because its children may have
different active time slots. Moreover, recall that a parent is allowed to transmit only
when its children are awake. For each layer $i$, where $0 < i \leq l$, BS-1 assigns a
minimal transmission time $t$ to the parent $P(v)$ of node $v \in L_i$ according to the
size of $|C(P(v), i)|$ in a non-increasing order manner (line 5 in Algorithm 3.3). That
is, parents with a high number of children will be scheduled first. For example in
Figure 3.2, for layer 2, recall that $|C(v_2, 2)| = 3$ and $|C(v_1, 2)| = 1$, and thus node
$v_2$ will be first scheduled to send the message to its children. Once a parent node $u$
is considered, it will be assigned with the minimum transmission time $t$ which is the
wake-up time of its children, and ensures the transmission from node $u$ is collision-
free. Let $u$ be the parent node of nodes in $L_i$ and the active time slot of nodes in $L_i$
is $\tau_i$. The minimum transmission time $t$ of $u$, i.e., $tr(u, i)$, must satisfy the following
constraints; see Algorithm 3.2,

1. The value of $t \mod T$ must equal $\tau_i$ (line 5); That is, $t \mod T$ must be an active
time slot of node $u$’s children;

2. Node $u$ must have received the message collision-free before time slot $t$ (line
5);

3. Nodes in $C(u, i)$ are not within the transmission range of any transmitting
nodes at time slot $t$ (line 2);

4. The reception time of nodes in $N_1(u) \setminus C(u, i)$ is not $t$ (line 3); That is, node $u$’s
one-hop neighbors $N_1(u)$ must not receive a message from their parent nodes
except node $u$’s children at $t$.

In other words, Algorithm 3.2 picks the minimum transmission time that coincides
with the active time of nodes in layer $i$, and avoids all transmission times of inter-
fering nodes. For any transmitting node $u$, Algorithm 3.2 records in the set $I_1(u)$
the transmission time of nodes that are in the transmission range of nodes in $C(u, i)$,
Algorithm 3.2 Schedule Transmission Time

1: Procedure MiniTransTime \( (u, \tau_i, T_b, G) \)
2: \( I_1(u) \leftarrow \{t | \exists w \in C(u, i) \text{ that hears a message at } t \} \)
3: \( I_2(u) \leftarrow \{t | \exists w \in N_1(u) \setminus C(u, i) \text{ that is scheduled to receive at } t \} \)
4: \( I(u) \leftarrow I_1(u) \cup I_2(u) \)
5: \( tr(u, i) \leftarrow \min\{t | t \mod T = \tau_i, t > rec(u) \text{ and } t \notin I(u)\} \)
6: return \( tr(u, i) \)

and likewise records in the set \( I_2(u) \) all reception time of nodes in \( N_1(u) \setminus C(u, i) \).

Hence, the set \( I = I_1 \cup I_2 \) contains all “busy” time slots. Consequently, Algorithm 3.2 returns the minimum transmission time not in \( I \).

The operation of Algorithm 3.3 is illustrated using Figure 3.2. It starts from layer 1. Nodes \( v_1 \) and \( v_2 \) will receive the message from node \( s \) at time 0. Then, for layer 2, parent node \( v_2 \) is first considered because it has the maximum number of children in layer 2, i.e., \( C(v_2, 2) = \{v_4, v_5, v_6\} \). Recall that \( rec(v_2) = 0 \), and the active time slot of nodes in \( L_2 \) is 2, i.e., \( \tau_2 = 2 \). As per Algorithm 3.2, \( I(v_2) = I_1(v_2) = I_2(v_2) = \emptyset \); that is, none of node \( v_2 \)’s one-hop neighbors hear or receive a message from other nodes. Therefore, \( tr(v_2, 2) = \min\{t | t \mod 4 = 2, t > rec(v_2) \text{ and } t \notin I(v_2)\} = 2 \). Then, parent node \( v_1 \) is considered. Given that \( C(v_1, 2) = \{v_3\} \) and \( rec(v_1) = 0 \), node \( v_6 \in N_1(v_1) \setminus C(v_1, 2) \) receives a message from node \( v_2 \) at time 2, and thus \( I_2(v_1) = \{2\} \). None of node \( v_1 \)’s children hear a message before \( v_1 \) is considered, hence \( I_1(v_1) = \emptyset \). The transmission time of node \( v_1 \), i.e., \( tr(v_1, 2) \) will be set to 6 because \( tr(v_1, 2) = \min\{t | t \mod 4 = 2, t > rec(v_1) \text{ and } t \notin \{2\}\} \). The other nodes are also handled in a similar manner and the final result is shown in Figure 3.2. In this example, the broadcast latency is 7. Note that, the reception time of nodes in layer \( L_3, L_4 \) and \( L_5 \) is smaller than that of node \( v_3 \) in layer 2; That is, BS-1 allows a node in a lower layer to transmit or receive a message earlier than a node in an upper layer.

3.2.1.1 Analysis

Theorem 3.1. Algorithm BS-1 provides a correct and collision-free broadcast schedule.

Theorem 3.2. BS-1 is an \((\Delta - 1)T\)-approximate solution for the MLBSDC problem.

Theorem 3.3. The total number of transmissions scheduled by BS-1 does not exceed
Algorithm 3.3 Broadcast Scheduling for BS-1

1: \( tr(s, 1) \leftarrow 0 \)
2: \( rec(v) \leftarrow 0, \forall v \in V \)
3: \textbf{for} \( i \leftarrow 2 \) to \( l \) \textbf{do}
4: \( \tau_i \leftarrow \) the active time slot of nodes in layer \( i \)
5: \( Q_i \leftarrow \) parents of nodes in layer \( i \) listed in the order when they were chosen as per line 7 in Algorithm 3.1
6: \textbf{while} \( Q_i \neq \emptyset \) \textbf{do}
7: \( q \leftarrow \) first node in \( Q_i \)
8: \( tr(q, i) \leftarrow \text{MiniTransTime} (q, \tau_i, T_b, G) \)
9: \( rec(w) \leftarrow tr(q, i), \forall w \in C(q, i) \)
10: \( Q_i \leftarrow Q_i \setminus \{q\} \)
11: \textbf{end while}
12: \textbf{end for}
13: \textbf{return} \( tr, rec \)

\(|V| - 1.\)

For complete proofs, please refer to Section A.1 and A.2 in Appendix.

3.2.2 BS-2

In this section, the second solution, BS-2, is detailed for the MLBSDC problem. BS-2 differs to BS-1 in terms of how parent nodes are chosen and how transmissions are scheduled. For instance, BS-2 constructs MIS to help determine parent-children relationship. In addition, BS-2 divides the schedule of parent nodes into two phases to reduce broadcast latency. Furthermore, BS-2 also schedules the transmissions of nodes based on SPT and has the same feature as BS-1, i.e., nodes in lower layers are allowed to transmit or receive earlier than nodes in upper layers.

Similar to BS-1, BS-2 starts by constructing \( T_{SPT} \) using Dijkstra’s algorithm. After that, it uses Algorithm 3.4 to construct the broadcast tree \( T_b \). As an example, the resulting \( T_b \) for the topology of Figure 3.1 is shown in Figure 3.3. As mentioned earlier, BS-2 uses MIS and a subset of dominated nodes or secondary nodes to broadcast a message. The following describes how these two sets of nodes are constructed.
Let $U$ denote the MIS of $G$, and $U_i$ is the MIS induced by nodes of layer $i$. Algorithm 3.4 constructs $U$ layer by layer; see line 5 to 9 in Algorithm 3.4. Specifically, it selects nodes at each layer that are not adjacent to nodes in $U$ greedily. Note that $U_0 = \{s\}$ and $U_1 = \emptyset$, because source node $s$ is the first node to be considered and all nodes in $L_1$ are adjacent to source node $s$. Moreover, BS-2 partitions nodes in $L_i$ into two subsets: $U_i$ and $M_i$, where $M_i = L_i \setminus U_i$; as an example, referring to Figure 3.3, for layer 2, it thus has $U_2 = \{v_3, v_4, v_5\}$ and $M_2 = \{v_6\}$. Globally, Let $M$ be $V \setminus U$, and hence, $M = \bigcup_{i=0}^{1} M_i$. Note that, nodes in $M$ and $M_i$ are also referred to as dominated nodes, and correspondingly, those in $U$ and $U_i$ are called dominators.

Once $U_i$ and $M_i$ are got, the next step is to determine the parent-children relationship in $T_b$ layer by layer in a top-down manner. Specifically, for each layer $i$, the algorithm first assigns a parent for nodes in $U_i$. It does this by preferentially selecting a parent $u$ in $M_{j<i}$ that covers the most nodes in $U_i$ that have yet to be assigned a parent (line 13 in Algorithm 3.4). As a result, the children of parent $u$ are all the dominators in $N_1(u) \cap U_i$ that do not have a parent yet; see line 14 of Algorithm 3.4. For example, referring to Figure 3.3, for nodes in $U_2 = \{v_3, v_4, v_5\}$, we see that node $v_1$ from layer 1 only covers node $v_3$, and node $v_2$ covers node $v_4$ and $v_5$, i.e., $|N_1(v_1) \cap U_2| = 1$ and $|N_1(v_2) \cap U_2| = 2$. Therefore, node $v_3$ is first considered as the parent of nodes $v_4$ and $v_5$, i.e., $P(v_1) = P(v_5) = v_2$ and $C(v_2, 2) = \{v_4, v_5\}$. Similarly, node $v_1$ is then
Algorithm 3.4 Broadcast Tree $T_b$ Construction for BS-2

1: $T_b \leftarrow (V_b, E_b), V_b \leftarrow V, E_b \leftarrow \emptyset$
2: $U \leftarrow U_0 \leftarrow \{s\}$
3: $P(v) \leftarrow \text{NIL}, \forall v \in V$
4: for $i \leftarrow 1$ to $l$ do
  5:   for each $w \in L_i$ do
  6:     if $(U \cap N_1(w)) = \emptyset$ then
  7:       $U_i \leftarrow U_i \cup \{w\}; U \leftarrow U \cup \{w\}$
  8:     end if
  9:   end for
10: $M_i = L_i \setminus U_i$
11: $U'_i \leftarrow U_i$
12: while $U'_i \neq \emptyset$ do
13:   $u \leftarrow$ a node in $M_{j<i}$ with the maximum
14:   $|\{v \mid v \in N_1(u) \cap U_i \text{ and } P(v) = \text{NIL}\}|$ value
15:   $C(u, i) \leftarrow \{v \mid v \in N_1(u) \cap U_i \text{ and } P(v) = \text{NIL}\}$
16:   $P(v) \leftarrow u$ and $U'_i \leftarrow U'_i \setminus \{v\}, \forall v \in C(u, i)$
17: end while
18: $M'_i \leftarrow M_i$
19: while $M'_i \neq \emptyset$ do
20:   $u \leftarrow$ a node in $U_{j\leq i}$ with the maximum
21:   $|\{v \mid v \in N_1(u) \cap M_i \text{ and } P(v) = \text{NIL}\}|$ value
22:   $C(u, i) \leftarrow \{v \mid v \in N_1(u) \cap M_i \text{ and } P(v) = \text{NIL}\}$
23:   $P(v) \leftarrow u$ and $M'_i \leftarrow M'_i \setminus \{v\}, \forall v \in C(u, i)$
24: end while
25: $E_b \leftarrow \{(u, v) \mid u = P(v)\}$
26: return $T_b = (V_b, E_b)$
chosen as the parent of node $v_3$, i.e., $P(v_3) = v_1$ and $C(v_1, 2) = \{v_3\}$.

After all nodes in $U_i$ are assigned a parent, the parent of nodes in $M_i$ are considered in the same way; see line 19 of Algorithm 3.4. These parent nodes of nodes in $M_i$ are chosen from $U_j$, where $j \leq i$. For instance, in Figure 3.3, node $v_6 \in M_2$ is only covered by dominator $s$, so $s$ is chosen as the parent of node $v_6$. The other layers are handled in a similar way and the final result is shown in Figure 3.3. Note that, the parent of nodes in $U_i$ (respectively, $M_i$) is chosen from the set $M_{j<i}$ (respectively, $U_{j\leq i}$) and all dominators in $U$ and their parent nodes form a CDS.

After constructing $T_b$, BS-2 schedules transmissions from parent nodes in $T_b$ as per Algorithm 3.5. Specifically, it consists of two phases. In Phase 1, the algorithm only considers the set $X$ comprising of nodes in $U$ and non-leaf nodes of $M$; see line 2 in Algorithm 3.5. That is, in Phase 1, this algorithm only schedules transmissions amongst nodes in the CDS, i.e., all dominators in $U$ and their parent nodes (or non-leaf nodes in $M$). In Phase 2, only dominators in $U$ send the message to all other nodes that are not assigned a reception time in Phase 1 (denoted by $Y$), see line 3 in Algorithm 3.5; That is, it only schedules the transmissions from nodes in the CDS to all other nodes. The rationale for having two phases is that it is not necessary to send a message to leaf nodes early as they are not responsible for relaying the message further. On the other hand, by reducing the number of receiving nodes in Phase 1, a transmitter will avoid a number of potential conflicts when sending a message to non-leaf nodes, thus reducing the broadcast latency.

Denote by $tr_1(v, i)$ and $tr_2(v, i)$ the scheduled transmission time at which node $v$ sends a message to its children in layer $i$ in Phase 1 and 2 respectively. In Phase 1, the scheduling starts at time slot 0 and work layer by layer in the top-down manner. Specifically, for each layer $i$, Algorithm 3.5 first schedules the parent nodes of dominators in $U_i$. Then, the transmissions of the parent of nodes in $M_i \cap X$ are scheduled after all nodes in $U_i$ are scheduled to receive the message. While scheduling transmissions, the parent of nodes in $U_i$ as well as $M_i \cap X$ are considered in the non-increasing order of the number of their children in $T_b$ (line 6 and 7 in Algorithm 3.5). For example, in layer 2 of Figure 3.3, parent node $v_1$ and $v_2$ will be scheduled before source node $s$ because node $v_1$ and $v_2$’ children $v_3$, $v_4$ and $v_5$ belong to
In Phase 2, transmissions are scheduled so that nodes in $Y = V \setminus X$ will receive the message. It also works layer by layer in the top-down manner. For each layer $i$, if node $w$ is not considered in Phase 1, i.e., $w \in Y$, the algorithm will apply Algorithm 3.2 to schedule its parent node $P(w)$ with the minimal transmission time $t$ in the same order as Phase 1 (i.e., parent node $P(w)$ with the maximum size of $C(P(w), i)$ will be scheduled first). For instance, in $T_b$ of Figure 3.3, $Y = \{v_9, v_{10}\}$. Node $v_9$ is the only node in $Y$ for layer 3. Hence its parent node $v_5$ will be scheduled to transmit to $v_9$ after Phase 1. Similarly, for layer 4, node $v_7$ is responsible for transmitting the message to $v_{10}$ after node $v_9$ is scheduled to receive.

Figure 3.3 is used to illustrate the operation of Algorithm 3.5. It starts with Phase 1, and set $X$ has members $\{s, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_{11}\}$. For layer 1, node $v_1$ and $v_2$ obtain their message from source $s$ at time slot 0, and therefore $t_{r1}(s, 1) = 0$. For layer 2, the transmission from node $v_2$ to node $v_4$ and $v_5$ in layer 2 is first scheduled because node $v_2$’s children $v_4$ and $v_5$ belong to $U_2$ and node $v_2$ has the most children in layer 2. As per Algorithm 3.2, none of node $v_2$’s one-hop neighbors hear or receive a message from other nodes when $v_2$ is considered, i.e., $I(v_2) = I_1(v_2) = I_2(v_2) = \emptyset$. Given that $r_{c}(v_2) = 0$ and $\tau_2 = 2$, and thus $t_{r1}(v_2, 2) = \min\{t \mid t \mod 4 = 2, t > 0\} = 2$. Then, parent node $v_1$ is considered. Recall that $C(v_1, 2) = \{v_3\}$ and $r_{c}(v_1) = 0$, none of node $v_1$’s one-hop neighbors hear or receive a message from node $s$ and $v_2$ when $v_1$ is considered, i.e., $I(v_1) = I_1(v_1) = I_2(v_1) = \emptyset$. Therefore, node $v_1$’s transmission time is also set to 2, i.e., $t_{r1}(v_1, 2) = 2$. Parent node $s$ is the last node to be considered for layer 2. Recall that $C(s, 2) = \{v_6\}$, node $v_6$ hears a message from node $v_1$ and $v_2$ when both of them transmit at time slot 2, and thus, $I_1(s)$ is set to $\{2\}$. Moreover, its one-hop neighbor $v_1$ and $v_2$ receive the message at time 0, i.e., $I_2(s) = \{0\}$. By Algorithm 3.2, $t_{r1}(s, 2) = \min\{t \mid t \mod 4 = 2, t > 0 \text{ and } t \notin I_1(s) \cup I_2(s)\} = 6$. All other layers are handled
Algorithm 3.5 Broadcast Scheduling for BS-2

1: \( tr_1(s, 1) \leftarrow 0 \) and \( rec(v) \leftarrow 0, \forall v \in V \)
2: \( X \leftarrow U \cup \{ w \mid w \in M \text{ and } w \text{ is a non-leaf node in } T_b \} \)
3: \( Y \leftarrow V \setminus X \)
   // Phase 1- schedule nodes in \( X \)
4: \textbf{for} \( i \leftarrow 1 \) to \( l \) \textbf{do}
5: \( \tau_i \leftarrow \) the active time slot of nodes in layer \( i \)
6: \( Q_{i1} \leftarrow \) parents of nodes in \( U_i \) with the order they were chosen as per line 13 of Algorithm 3.4.
7: \( Q_{i2} \leftarrow \) parents of nodes in \( M_i \cap X \) with the order they were chosen as per line 19 of Algorithm 3.4.
8: \( Q_i \leftarrow Q_{i1} \cup Q_{i2} \) // nodes in \( Q_{i1} \) ordered before \( Q_{i2} \)
9: \textbf{while} \( Q_i \not= \emptyset \) \textbf{do}
10: \( q \leftarrow \) first node in \( Q_i \)
11: \( tr_1(q, i) \leftarrow \text{MiniTransTime} \left( q, \tau_i, T_b, G \right) \)
12: \( rec(w) \leftarrow tr_1(q, i), \forall w \in C(q, i) \cap X \)
13: \( Q_i \leftarrow Q_i \setminus \{ q \} \)
14: \textbf{end while}
15: \textbf{end for}
   // Phase 2- schedule nodes in \( U \cup Y \)
16: \textbf{for} \( i \leftarrow 1 \) to \( l \) \textbf{do}
17: \textbf{if} \( v \in (M_i \cap Y) \) \textbf{then}
18: \( \tau \leftarrow \) the active time slot of nodes in layer \( i \)
19: \( Q_i \leftarrow \) parents of nodes in \( M_i \cap Y \) with the order they are chosen in line 19 of Algorithm 3.4
20: \textbf{while} \( Q_i \not= \emptyset \) \textbf{do}
21: \( q \leftarrow \) first node in \( Q_i \)
22: \( tr_2(q, i) \leftarrow \text{MiniTransTime} \left( q, \tau_i, T_b, G \right) \)
23: \( rec(w) \leftarrow tr_2(q, i), \forall w \in C(q, i) \cap Y \)
24: \( Q_i \leftarrow Q_i \setminus \{ q \} \)
25: \textbf{end while}
26: \textbf{end if}
27: \textbf{end for}
28: \textbf{return} \( tr_1, tr_2 \) and \( rec \)
in a similar way until all nodes in $X$ are scheduled to receive the message collision-free.

In Phase 2, set $Y$ has members $\{v_9, v_{10}\}$. Algorithm 3.5 starts with layer 3. Given that $P(v_9) = v_5$, $rec(v_5) = 2$ and $\tau_3 = 3$, node $v_9$ does not hear any message from its one-hop neighbor $v_5$; That is, in Phase 2, $I_1(v_5) = \emptyset$. Node $v_2$, which is node $v_5$’s one-hop neighbor, receives the message at time 0, and thus $I_2(v_5) = \{0\}$. By Algorithm 3.2, $tr2(v_5, 3) = 3$, i.e., $\min \{t \mid t \mod 4 = 3, t > 2 \text{ and } t \notin \{0\}\} = 3$. Similarly, for layer 4, node $v_{10}$’s parent node $v_7$ is scheduled to transmit at time 4, i.e., $rec(v_7) = 3$, $\tau_4 = 0$ and $I(v_7) = \{0\}$, where $I_1(v_7) = \emptyset$ and $I_2(v_7) = \{0\}$. The final result is shown in Figure 3.3. In this example, BS-2 takes 10 time slots to finish the broadcast. Note that, $tr1(v_7, 4) = 4$, $tr1(s, 2) = 6$, and node $v_7$, $s$ lie in layer 3 and 0 respectively. Clearly, the transmission from node $v_7$ to $v_{10}$ is earlier than that from $s$ to $v_6$. It means BS-2 also allows a node in a lower layer to receive or transmit a message earlier than a node in an upper layer.

### 3.2.2.1 Analysis

**Theorem 3.4.** BS-2 yields a correct and collision-free broadcast schedule.

**Theorem 3.5.** BS-2 provides a $13T$-approximate solution for the latency.

**Theorem 3.6.** BS-2 is a $4(T + 3)$-approximate solution in terms of number of transmissions.

For complete proofs, please refer to Section A.3 and A.4 in Appendix.

### 3.3 Evaluation

Herein, this section outlines the research methodology used to evaluate the performance of BS-1 and BS-2. This section compares them against SPT, i.e., Dijkstra algorithm, and OTAB [46], which is known to have the lowest constant approximation ratio to date. Note that SPT can be used to obtain the lower bound for broadcast latency, assuming no collisions. The main goal of simulations in this section is to compare the theoretical and experimental performance of proposed algorithms in this
chapter. To this effect, the designed experiments focus on the effect of various network configurations on proposed algorithms’ performance, and do not employ packet level simulations. These experiments measure each algorithm against the following metrics:

- *Broadcast latency* – the total time required by all nodes to receive a broadcast message; and

- *Transmission ratio* – the ratio between the number of times a broadcast message is transmitted and number of nodes in the network.

These experiments are conducted in MATLAB [57]. All nodes are stationary and randomly deployed in a square area of $200 \times 200$ m$^2$. These experiments will be used to study the effect of different network configurations including network size, transmission radius and duty cycle. The network size ranges from 200 to 1000 with an interval of 200. The transmission radius ranges from 20 to 60 meters. The duty cycle is defined as $\frac{1}{T}$ and varies from 0.1 to 0.02. Every experiment is conducted with one change to the network configuration whilst the other fixed. Each experiment is conducted on 20 randomly generated topologies. Moreover, for each topology, it carries out the experiment for 10 runs, and in each run, an arbitrary node is selected as the source node. Each result is the average of 200 simulation runs. As mentioned earlier, the broadcast algorithms designed in this chapter do not apply any specific asynchronous pseudo-random MAC protocols. The focus is on scheduling the broadcast of a message as per the active time slots of nodes, which incurs a much higher delay relative to the transmission time [33].

### 3.3.1 Impact of Network Size

The first experiment reports on the impact of network size. In this experiment, the transmission radius is fixed at 30 m and the duty cycle is set to 0.05. As shown in Figure 3.4, we can find that the broadcast latency of all algorithms except SPT grows with increasing network sizes because more nodes will need to receive the broadcast message and these algorithms are required to avoid more collisions. BS-1 and BS-2 perform better than OTAB, i.e., when the network size is set to 1000, the
broadcast latency of BS-1 is only $\frac{1}{17}$ of that of OTAB, because BS-1 and BS-2 allow nodes in lower layers to receive or transmit earlier than nodes in upper layers, and thus reducing the broadcast latency. Moreover, BS-1’s performance is better than that of BS-2 before the network size reaches 300, but after that, BS-2 performs better than BS-1. This is because the broadcast latency of BS-1 is bounded by $\Delta$. With increasing network size, $\Delta$ will become very large; in contrast, BS-2’s broadcast latency is mainly influenced by a constant number, i.e., 13. Additionally, Figure 3.4 also shows that BS-1, BS-2 and OTAB’s broadcast latency is bounded by that of SPT.

![Figure 3.4: Broadcast latency under different network sizes](image)

From Figure 3.5, we observe that the transmission ratio for all algorithms decreases when the network size increases. This is mainly because the average degree grows with increasing the network size, thereby, allowing one transmission to reach more nodes; That is, on average, each node needs fewer transmissions to inform its one-hop neighbors. Moreover, BS-1 and BS-2 perform better than OTAB in terms of transmission ratio. Among all algorithms, BS-1 outputs the smallest transmission ratio; i.e., when the network size is set to 1000, BS-1 achieves a 60% improvement in terms of transmission ratio as compared to OTAB.
3.3.2 Impact of Transmission Radius

Next, this simulation evaluates the performance of all algorithms under different transmission radii. The network size is set to 400 and the duty cycle of nodes is fixed at 0.05. As shown in Figure 3.6, the broadcast latency of all algorithms decreases with increasing transmission radius. This is because broadcast latency is mostly influenced by the number of layers in the SPT, and nodes with a larger transmission radius will have more neighbors which helps to reduce the number of layers in the SPT. Moreover BS-1 and BS-2 perform better than OTAB in terms of broadcast latency for the same reason as listed in section 3.3.1, i.e., their latency is within 6% of the latency achieved by OTAB.

Figure 3.7 shows that the transmission ratio for BS-1, BS-2 and OTAB also decreases as the transmission radius grows. When the transmission radius increases, a transmitting node can inform more nodes, and thus fewer nodes will be needed to retransmit a broadcast message. This leads to a decline in transmission ratio. Furthermore, BS-1 has the best performance in terms of transmission ratio; specifically, about 40% better than that OTAB when the transmission radius is 60 m.
Figure 3.6: Broadcast latency under different transmission radii

Figure 3.7: Transmission ratio under different transmission radii
3.3.3 Impact of Duty Cycle

Finally, the impact of duty cycle on the performance of all algorithms is studied. The network size is fixed to 400 and the transmission radius is set to 20 m. From Figure 3.8, we find that with declining duty cycle, the broadcast latency for all algorithms increases. The scheduling period $T$ contains more time slots as duty cycle decreases, so there will be more layers in the SPT, leading to increasing broadcast latencies for all algorithms that schedule their transmissions based on the SPT. In addition, BS-1 and BS-2 perform better than OTAB, i.e., BS-1’s broadcast latency is around $\frac{1}{15}$ of that of OTAB when the duty cycle is set as 0.02.

![Figure 3.8: Broadcast latency under different duty cycles](image)

Figure 3.9 shows that the transmission ratio for all algorithms increases with decreasing duty cycle. This is because when the duty cycle decreases, the scheduling period $T$ will contain more time slots. As a result, a transmitting node may need to transmit a number of times because its neighbors have a higher probability of choosing different active time slots. Moreover, BS-1 outputs the best transmission ratio, and BS-2 and OTAB have a similar performance in terms of transmission ratio.
3.4 Remarks on Duty Cycle and Unreliable Links

Without loss of generality, BS-1 and BS-2 assume each node is only active for one time slot, and at this time, it transmits one message. In fact, a node may choose to be active in more than one time slot. In the extreme case, a node may be awake in all slots, thus creating an always-on WSN. BS-1 and BS-2 can be adapted as follows to support any multi-slot case: (1) $\tau(v)$ and $\tau_i$ should be defined as the first active time slot in the period $T$ of node $v$ or nodes in layer $i$ respectively, and (2) accordingly, the first constraint of Algorithm 3.2 is changed as follows: the value of $t \mod T$ must equal to one of the active time slots of nodes in layer $i$.

BS-1 and BS-2 are also applicable when considering unreliable links. As shown in Section 3.1.1, assume a message can be successfully delivered from a sender to a receiver within a time slot. In reality, as shown in [33], the maximum size of a typical TinyOS packet is 47 bytes, a time slot is usually set to 20 ms, and thus, a MicaZ node can attempt at least 13 transmissions in one time slot. In other words, although low-power wireless links are generally unreliable, it can ensure that messages can be successfully transmitted within a time slot through multiple transmission times.
3.5 Conclusion

This chapter has presented a study on the minimum latency broadcast scheduling problem in the context of duty-cycled WSNs. The main difficulty is that sensor nodes are not synchronized and are not awake simultaneously. To overcome this problem, this chapter outlines two novel algorithms: BS-1 and BS-2, and proves conclusively that these algorithms provide correct and collision-free schedules, and produce low broadcast latency and low overheads. Furthermore, BS-2 achieves the best constant approximation ratio for latency amongst the algorithms proposed for the MLBSDC problem; see Table 2.4. The simulation results show that BS-1 and BS-2 have better performance in terms of broadcast latency and transmission redundancy than OTAB [46] under different network scenarios.

However, BS-1 and BS-2 are both centralised algorithms. To design a distributed algorithm for the MLBS problem under the RTS/CTS interference model, the next chapter outlines a greedy heuristic algorithm and its distributed implementation. The chapter also shows that the centralised version has a constant approximation ratio in terms of the broadcast latency.
Centralised and Distributed Algorithms under the RTS/CTS Interference Model

Asynchronous duty-cycled MAC protocols do not require global synchronization because nodes determine their wake-up schedule independently and randomly. As a result, these MACs have superior performance to those that employ synchronous duty-cycles in terms of energy expenditure, and advantageously, they are simple to implement. A key limitation is that they do not support efficient broadcast. A node needs to transmit a broadcast packet multiple times because only a subset of its neighbors may be awake at any given point in time. To address the minimum broadcast latency problem in duty-cycled WSNs (MLBSDC), Chapter 3 presented two centralised approximation algorithms, which rely on a broadcast tree, and proved these two algorithms are constant approximation solutions. However, as shown in Table 2.4, it remains an open problem in general to implement these tree-based algorithms for the MLBSDC problem in a distributed manner.

Henceforth, this chapter outlines a greedy heuristic algorithm and its distributed implementation, CEN and DIS respectively, under the RTS/CTS interference model, where CEN produces an approximation ratio of $(\Delta - 1)T$ in terms of broadcast latency. Here, $\Delta$ and $T$ are the maximum degree and maximum interval between two adjacent broadcast time slots for a node in a given WSN respectively. In particular, for always-on networks, i.e., $T = 1$, the approximation ratio is $\Delta - 1$. In addition, it uses a novel asynchronous MAC protocol, i.e., ArDeZ [15], which en-
sures all neighbors of a broadcasting node are awake to receive a broadcast. The performance of these proposed algorithms is evaluated under different network configurations. Extensive simulation studies show that CEN and DIS have near optimal network performance in terms of broadcast latency. In particular, compared to [46], the best broadcast scheduling algorithm to date, the broadcast latency and number of transmissions achieved by CEN are $\frac{1}{5}$ and $\frac{1}{2}$ of that of [46] on average, respectively.

4.1 Preliminaries

In this chapter, the network model is first introduced, followed by the key properties of a pseudo-random asynchronous MAC, i.e., ArDeZ [15].

4.1.1 Network Model

Similar to Chapter 3, the WSN is modelled as a UDG, $G = (V, E)$, where $V$ denotes the set of nodes and $E$ denotes the set of links that exist between two nodes if their Euclidean distance is no more than the unit transmission range $r_T$. Let $N_1(i)$ denote the set of one-hop neighbors of node $i$. Accordingly, $N_2(i)$ denotes the two-hop neighbors of $i$. There are no packet or bit errors and links are bidirectional. However, this chapter assumes the RTS/CTS interference model, meaning a packet is considered lost if there is a collision; i.e., given a receiving node $i$, more than two nodes in $N_1(i)$ transmit simultaneously. Apart from that, time is discrete, whereby it is divided into fixed time slots of equal length, denoted by $0, 1, 2, 3, \cdots$. Each time slot is assumed to be of sufficient duration to receive a message. The network is locally synchronized at a slot level. As shown in [15] and [33], this can be achieved if each node keeps the clock drift information of its neighbors. Furthermore, transmissions are not required to start at the beginning of each slot, meaning nodes do not need strict synchronization in order to communicate. Without loss of generality, it is assumed that the clock drift between any two nodes is zero. In a given time slot, a node is either in active or a dormant state. A node is able to transmit or receive a packet in its active state, but a dormant node will switch all its components off except for a wake-up timer.
4.1.2 ArDez MAC Model

Different from conventional duty-cycle-aware MAC models, for example those used by works in Section 2.3.3, where a node transmits a message at the corresponding receiver’s wake-up slot, in this chapter, all nodes determine their own working schedule independently using ArDeZ [15] – a low power asymmetric rendezvous MAC protocol. As shown in [15], ArDeZ operates under two scenarios: unicast and broadcast. In the unicast setting, because each node is aware of its neighbors’ pseudo-random wake-up times, nodes will send a message only when its corresponding receiver wakes up. For broadcast, each node is assigned a broadcast time that is generated pseudo randomly, meaning as long as a node is aware of its neighbor’s broadcast seed (or ID), which is exchanged periodically, it will be able to determine the neighbor’s broadcast times. At these times, the node will wake-up to receive any broadcast message from its neighbors.

This chapter models ArDeZ’s operation under the broadcast scenario as follows. Let $T$ be the maximum interval between two adjacent broadcast time slots for a node. Given a node $i$, its broadcast time slot is determined by the output of its Pseudo-Random Number Generator (PRG) modulo $T$; i.e., yielding a series of random integer numbers in the range $[1, 2, \cdots, T]$. For a given node $i$, its $n$-th random integer and broadcast time slot is denoted by $Rand(i, n)$ and $Tx(i, n)$ respectively. Hence, $Tx(i, n)$ can be calculated as follows,

$$Tx(i, n) = \sum_{j=1}^{n} Rand(i, n) \quad (4.1)$$

All of node $i$’s one-hop neighbors, i.e., $N_1(i)$, are aware of node $i$’s ID and can also compute node $i$’s broadcast time slot by running Equ. 4.1. Therefore, at time slot $Tx(i, m)$, all nodes in $N_1(i)$ will wake-up to receive a broadcast packet from $i$. Note that nodes are able to receive packets at any of its neighbors’ wake-up time slots, but are only allowed to transmit in their respective broadcast time slot – as determined by Equ. 4.1. As an aside, in order to reduce idle listening, a node will send a short preamble if a transmission is forthcoming. Upon detecting this preamble,
a node remains awake and prepares to receive the message. Otherwise, it goes back to sleep immediately. The duty cycle of a node is defined as \( \frac{1}{E_P} \), where the numerator corresponds to one active time slot, and \( E_P \) is the expected value of the PRG, which is equal to \( T_2 \) when PRG satisfies a uniform distribution.

The advantage of using ArDeZ is that it does not require nodes to remain awake or wait idly for a receiver to wake up. In particular, nodes wake up simultaneously to receive from a given neighbor, and thereby, take advantage of the inherent broadcast nature of the wireless channel as opposed to conventional duty-cycle-aware MACs such as [50] and [74], which require a node to transmit multiple times to inform its neighbors during broadcast. More importantly, it retains the benefits of asynchronous duty cycle as there is no need for a global clock synchronization protocol. Hence, compared with conventional asynchronous MAC models, ArDez is capable of achieving shorter broadcast latency and smaller transmission times. As will be shown in Section 4.3, the performance of broadcast algorithm under ArDeZ is much better than using a conventional MAC, as demonstrated by its shorter broadcast latencies and fewer transmission times. A summary of notations used in this chapter can be found in Table 4.1.

<table>
<thead>
<tr>
<th>( G(V, E) )</th>
<th>Network graph</th>
<th>( Lat )</th>
<th>Broadcast latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N_1(v) )</td>
<td>( v )'s one-hop neighbors</td>
<td>( tr(v) )</td>
<td>( v )'s transmission time</td>
</tr>
<tr>
<td>( N_2(v) )</td>
<td>( v )'s two-hop neighbors</td>
<td>( rec(v) )</td>
<td>( v )'s reception time</td>
</tr>
<tr>
<td>( Tx(v,n) )</td>
<td>( v )'s ( n^{th} ) broadcast time</td>
<td>( t_{min} )</td>
<td>Minimum transmission time</td>
</tr>
<tr>
<td>( G_X )</td>
<td>Graph of MTS</td>
<td>( G_Y )</td>
<td>Graph of MITS</td>
</tr>
<tr>
<td>( X )</td>
<td>Set of nodes with ( t_{min} )</td>
<td>( N_X )</td>
<td>One-hop neighbors of ( X ) in ( G_X )</td>
</tr>
<tr>
<td>( Y )</td>
<td>Output of MTS</td>
<td>( N_Y(v) )</td>
<td>One-hop neighbors of ( v ) in ( G_Y )</td>
</tr>
<tr>
<td>( W(v) )</td>
<td>Weight of ( v ) in ( G_Y )</td>
<td>( d(v) )</td>
<td>Degree of ( v ) in ( G_Y )</td>
</tr>
<tr>
<td>( M_1(v) )</td>
<td>MyStatus of ( v )</td>
<td>( M_2(v) )</td>
<td>MyNeighbors of ( v )</td>
</tr>
<tr>
<td>( A )</td>
<td>( v )'s one-hop neighbors with ( TxBit=0 )</td>
<td>( B )</td>
<td>( v )'s two-hop neighbors with ( RxBit=0 )</td>
</tr>
<tr>
<td>( A_1 )</td>
<td>Nodes in ( A ) with ( RxBit=1 )</td>
<td>( A_0 )</td>
<td>Nodes in ( A ) with ( RxBit=0 )</td>
</tr>
<tr>
<td>( A_1 )</td>
<td>Nodes in ( A_1 ) with ( t_{min} )</td>
<td>( A_0 )</td>
<td>Nodes in ( A_0 ) with ( t_{min} )</td>
</tr>
<tr>
<td>( T )</td>
<td>Maximum wake-up interval</td>
<td>( \Delta )</td>
<td>Maximum degree of ( G )</td>
</tr>
</tbody>
</table>
4.2 Proposed Algorithms

This section first outlines the optimal algorithm used to compute the lowest broadcast latency bound for a given network. Note, however, it does not consider collisions. After that, Section 4.2.2 and 4.2.3 expound centralized and distributed broadcast algorithms.

4.2.1 Optimal (OPT) Broadcast Algorithm

In OPT, the Dijkstra algorithm is adapted to consider the wake-up times of nodes. Let \( tr(i) \) and \( rec(i) \) denote node \( i \)'s transmission and reception time respectively. Every node in the network can be in one of three states: \textit{unvisited}, \textit{current}, \textit{visited}. A node that has not been considered by OPT is in the \textit{unvisited} state. On the other hand, a \textit{visited} node has been considered, and OPT will not check it again; i.e., its reception time is final and minimal. Lastly, a node that is under consideration is marked \textit{current}. OPT proceeds as follows:

1. Assign a reception time of \( t_0 \) to the source node, and \( \infty \) as the transmission and reception time of all other nodes. Assign the source node to the set of \textit{current} nodes \( U \), which is initially empty.

2. Mark all nodes as \textit{unvisited}, except for the source node.

3. For each node \( i \in U \), update node \( i \)'s transmission time \( tr(i) \) to be

\[
\min \{ t \mid t > rec(i) \text{ and } t = Tx(i,k) \} \tag{4.2}
\]

where \( k \in \mathbb{N} \). That is, choose the next active time slot that follows the reception time.

4. For each node \( i \in U \), iterate through all of node \( i \)'s \textit{unvisited} neighbors and update their receiving time as follows: if an \textit{unvisited} neighbor \( j \) satisfies \( tr(i) < rec(j) \), node \( j \)'s reception time \( rec(j) \) will be updated to \( tr(i) \) and its transmission time \( tr(j) \) will also be calculated as per Step-3; Otherwise, it remains unchanged.
5. Mark node $i$ as visited, and remove it from $U$. A visited node will not be checked ever again. Its receiving time recorded now is final and minimal.

6. If all nodes have been visited, finish. Otherwise, include unvisited node(s) with the smallest transmission time, denoted by $t_{\text{min}}$, into $U$ and continue from Step-4.

The lower bound of the broadcast latency $Lat$ is:

$$Lat = \max(\text{rec}(i)) - t_0 \quad \forall i \in V$$

(4.3)

Figure 4.1 is used as an example to illustrate the operation of OPT algorithm. The network consists of nine nodes as shown in Figure 4.1. Node $s$ is the source node and the starting time is $t_0 = 0$. Table 4.2 lists the first three pseudo-random broadcast time slots of all nodes. OPT first set $\text{rec}(s) = 1$, assign $\infty$ for all other nodes and add node $s$ into $U$ as per Step-1. Then, it marks all other nodes as unvisited. Next, it calculates the transmission time $tr(s)$ of node $s$ as per Step-3. As shown in Table 4.2, $Tx(s, 1) = 2$, $Tx(s, 2) = 4$ and $Tx(s, 3) = 11$, and hence, $tr(s) = 2$, i.e., $\min \{ t \mid t > 1 \text{ and } t = Tx(s, k) \}$, where $k = 1, 2, 3$. After that, for each unvisited neighbors of node $s$, namely, node $v_1$, $v_6$ and $v_7$, set their receive time $\text{rec}(v_1) = \text{rec}(v_6) = \text{rec}(v_7) = 2$, because their previous reception time $\infty$ is larger than $tr(s)$. Calculate the transmission time of node $v_1$, $v_6$, and $v_7$ as per Step-3. For instance, node $v_1$’s first three broadcast time slot is 1, 5 and 10, and thus its transmission time $tr(v_1)$ is $\min \{ 5, 10 \} = 5$. Next, mark node $s$ as visited and remove it from $U$ as per Step-5. Then, include node $v_1$, $v_6$, and $v_7$ into $U$ because they have the smallest transmission time, i.e., 5, among all unvisited nodes. The other nodes are handled in a similar manner until all nodes are visited and the final result is shown in Figure 4.1. In this example, the broadcast latency is 6, i.e., $Lat = \max\{1, 2, 5, 6\} - t_0 = 6$

4.2.2 Centralized (CEN) Broadcast Algorithm

Recall that OPT does not consider collision. In Figure 4.1, nodes $v_5$ and $v_8$ will experience a collision when nodes $v_1$, $v_6$ and $v_7$ transmit simultaneously to their common nodes $v_5$ and $v_8$ at time slot 5. Furthermore, node $v_6$ does not need to
Figure 4.1: An illustration of OPT. The label \( \text{rec, tr} \) denotes the reception and transmission time of a node.

Table 4.2
First three broadcast time slots of all nodes in Figure 4.1

<table>
<thead>
<tr>
<th>ID</th>
<th>s</th>
<th>( v_1 )</th>
<th>( v_2 )</th>
<th>( v_3 )</th>
<th>( v_4 )</th>
<th>( v_5 )</th>
<th>( v_6 )</th>
<th>( v_7 )</th>
<th>( v_8 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_x(ID, 1) )</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>( T_x(ID, 2) )</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>4</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>( T_x(ID, 3) )</td>
<td>11</td>
<td>8</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>10</td>
<td>9</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>

transmit the packet to node \( v_5 \) because node \( v_7 \), which covers more neighbors, is a better candidate to transmit the packet to node \( v_5 \) instead of node \( v_6 \). Therefore, to avoid collisions and reduce the number of transmissions, we want to schedule current nodes that share common neighbors to transmit at different time slot or stop transmission in Step 6 of OPT. Accordingly, Step-6 is modified as follows:

6. If all nodes have been visited, finish. Otherwise, apply the function: (i) Minimal Transmission Set (MTS), and (ii) Maximal Independent Transmissions Set (MITS), to find the next current nodes and include them in \( U \); continue from Step-4.

Next, both functions (i) MTS, and (ii) MITS, are defined precisely.

4.2.2.1 MTS

The aim of this function is to reduce redundant broadcasts, and consequently, extends network lifetime. The MTS function takes as inputs set \( X \) and set \( N_X \), and
outputs the Minimal Transmission Set, denoted by $MTS(X, N_X)$. The said inputs are defined as follows: $X$ contains unvisited nodes with the smallest transmission time of $t_{min}$, and $N_X$ denotes the unvisited one-hop neighbors of nodes in $X$.

MTS function starts by constructing a new sub graph $G_X$, which only contains nodes in $X$ and $N_X$. This new sub graph $G_X$ satisfies the following rules:

1. There is no edge between nodes in $X$ or $N_X$.
2. An edge exists between node $v \in X$ and node $u \in N_X$ if node $v$ and $u$ are in each other’s transmission range.

The MTS problem can be reduced to the Set Covering Problem, which is known to be NP-hard. Hence, once $G_X$ is constructed, a greedy algorithm, which is based on the Multi Point Relay (MPR) algorithm described in [66], is applied to calculate set $MTS(X, N_X)$ in $G_X$:

1. Start with an empty minimal transmission set: $MTS(X, N_X) = \{ \}$. Mark all nodes in $N_X$ as uncovered.
2. Identify nodes in $N_X$ that only have one neighbor in $X$. Mark these nodes as covered, and include their neighbors in $X$ into $MTS(X, N_X)$. Mark all nodes covered by nodes in $MTS(X, N_X)$ as covered.
3. While there are still uncovered nodes in $N_X$, find a node in $X$, which is not in $MTS(X, N_X)$ and covers the largest number of uncovered nodes in $N_X$ (break the ties on the basis of smaller ID). Include said node in $MTS(X, N_X)$, and mark the corresponding nodes in $N_X$ as covered. Repeat this step until all nodes in $N_X$ are covered.
4. Return $MTS(X, N_X)$. Note that, nodes in set $(X - MTS(X, N_X))$ are excluded from transmitting the packet.

The earlier example is modified to show how the aforementioned steps work. Recall that node $v_1$, $v_6$ and $v_7$ have the smallest transmission time of ’5’, i.e., $t_{min} = 5$; see
Figure 4.1. $G_X$ is constructed with $X = \{v_1, v_6, v_7\}$ and $N_X = \{v_2, v_3, v_5, v_8\}$, and the resulting graph $G_X$ is shown in Figure 4.2.

After the construction of $G_X$, MTS then carries out the following steps:

1. Set $MTS(X, N_X) = \{}$. Mark all nodes $N_X = \{v_2, v_8, v_3, v_5\}$ as uncovered.

2. Include node $v_1$ and $v_7$ into $MTS(X, N_X)$, because node $v_2$ and $v_3$ only have one node, $v_1$ and $v_7$ respectively, in $X$. Mark node $v_2$, $v_3$, $v_8$ and $v_5$ as covered.

3. In this step, as all nodes are covered, this step is skipped.

4. Return $MTS(X, N_X) = \{v_1, v_7\}$. Here, node $v_6$ is redundant as node $v_5$ will receive the packet from node $v_7$.

4.2.2.2 MITS

After finding $MTS(X, N_X)$, the next step is to ensure a collision-free transmission. That is, the goal is to segregate nodes with the same transmission time. This can be performed easily as any node’s broadcast time slot is aware; see Section 4.1. Let $Y$ denote $MTS(X, N_X)$. The MITS function accepts set $Y$ and set $N_X$ as inputs, and outputs $MITS(Y, N_X)$, a set containing nodes that are able to transmit at time $t_{\text{min}}$ concurrently.

This function also starts by constructing a conflict graph $G_Y$ as follows:

1. Vertices in $G_Y$ are nodes in $Y$. 
2. Each node in $G_Y$ has a weight $W(i)$ that corresponds to the number of neighbors in $G_X$.

3. An edge exists between node $v$ and $u$ in $G_Y$ if they have common neighbors in $G_X$.

The MITS problem can be induced to be the Maximal Weight Independent Set problem (MWIS), which is known to be NP-hard. A greedy algorithm, based on [67], is outlined as follows,

1. Start with $MITS(Y, N_X) = \{ \}$. 

2. Add a node $i$ into $MITS(Y, N_X)$ where the ratio of $W(i)/(d(i) + 1)$ is maximum (break the ties on the basis of smaller ID). Here $d(i)$ is the degree of node $i$ in $G_Y$. In other words, the node that has the highest coverage and least conflict is selected.

3. Define the set of one-hop neighbors of node $i$ in $G_Y$ as $N_Y(i)$. For each neighbor $j \in N_Y(i)$, update its transmission time $t_r(j)$ to

$$
\min \{ t \mid t > t_{min} \text{ and } t = T_x(j, k) \} \tag{4.4}
$$

where $k \in \mathbb{N}$.

4. Remove node $i$ and $N_Y(i)$ from $G_Y$.

5. Repeat from Step-2 until there are no nodes in $G_Y$.

6. Return $MITS(Y, N_X)$.

Figure 4.3 shows an example of MITS. After the completion of MTS in Figure 4.2, it has that $MTS(X, N_X) = \{ v_1, v_7 \}$. According to OPT, both nodes in $MTS(X, N_X)$ have a transmission time of 5. Unfortunately, they cannot transmit simultaneously as node $v_8$ is their common neighbor. As per MITS, it first constructs the conflict graph $G_Y$ using nodes in $MTS(X, N_X)$ and $N_X$. The resulting graph $G_Y$ is shown in Figure 4.3. The weight of each node is shown next to it. We see that node $v_1$ and $v_7$ have two and three neighbors in $G_X$ respectively; see Figure 4.2. MITS then proceeds as follows:
1. \( MITS(Y, N_X) = \{ \} \)

2. Add node \( v_7 \) to \( MITS(Y, N_X) \) as \( W(v_7)/(d(v_7) + 1) \) yields the maximum value of 1.5.

3. As \( N_Y(v_7) = \{ v_1 \} \), it updates the transmission time of node \( v_1 \), i.e., \( tr(v_1) \), to 8 as per Table 4.2.

4. Remove node \( v_7 \) and \( v_1 \) from \( G_Y \).

5. Finish, as there are no vertices in \( G_Y \).

6. Return \( MITS(Y, N_X) = \{ v_7 \} \) which will transmit at time slot \( t_{\text{min}} = 5 \).

After MTS and MITS, set \( U = MITS(Y, N_X) \), where node \( v_7 \) is the current node in the example, whereas node \( v_1 \) is still unvisited and its transmission time, i.e., \( tr(v_1) \), is set to 8, as per Table 4.2. Node \( v_6 \) is excluded from transmission as it is redundant. Repeat from Step-4 of OPT until all nodes are visited and the final result is shown in Figure 4.4. The broadcast latency for CEN in this example is 8 and the total number of transmissions is four, i.e., node \( s \), \( v_1 \), \( v_5 \) and \( v_7 \).

Figure 4.3: Graph \( G_Y \) with \( Y = \{ v_1, v_7 \} \) and \( N_X = \{ v_2, v_3, v_5, v_8 \} \)

Figure 4.4: An illustration of CEN
4.2.2.3 Discussion

The next theorems and lemmas assert the time complexity and correctness of CEN, and build an upper bound on the broadcast latency produced by CEN.

First define a shortest path tree $T_{SPT}$ as one that is rooted at source node $s$, and can be constructed according to the OPT algorithm, where an edge exists between node $i$ and node $j$ if node $i$ transmits the packet to node $j$ in OPT. Then, we divide nodes into different layers based on their minimal transmission time assigned by OPT, i.e., nodes in the same layer of $T_{SPT}$ share the same minimal transmission time. Let $L_k$ denote the set to nodes at layer $k$, e.g., $L_0 = \{s\}$, and $T(k)$ denote the minimal transmission time assigned by OPT to nodes in layer $k$, where $0 \leq k \leq l$. Here, $l$ is the maximum number of layers in $T_{SPT}$. As illustrated in Section 4.2.1, OPT yields the lower bound for broadcast latency, i.e., $T(l)$, where $T(l) \geq l$.

**Lemma 4.1.** The time complexity of OPT, MTS and MITS is $O(|V|^2)$.

**Proof.** In OPT, steps 4-6 are the most computationally intensive steps, and are repeated $|V|$ times. In particular, for each time, line 4 will iterate through all of the unvisited neighbors of a current node in $U$, and hence, for total $|V|$ times, line 4 will take $O(|E|)$ time, where $|E|$ is the number of edges in $G$. For step 5, it will take $O(|V|)$ time before OPT finishes. For each iteration of step 6, it requires $O(|V|)$ as the algorithm needs to iterate through $|V|$ nodes to find current nodes. For $|V|$ times, step 6 needs to take $O(|V|^2)$ time. The total time of steps 4-6 is upper-bounded by $O(|E|) + O(|V|) + O(|V|^2)$. Note, $|E| \leq 1/2 |V| (|V| - 1)$. Thus, OPT takes at most $O(|V|^2)$ time.

For MTS, let $n = |G_X|$, $n_1 = |X|$ and $n_2 = |N_X|$. After the construction of $G_X$, both step 1 and 2 require at most $O(n_2)$ time to mark nodes in $N_X$ as uncovered, and to identify nodes in $N_X$ which are only covered by a node in $X$. Lastly, step 3 takes at most $O(n_1^2)$ time to find nodes in $X$ which covers the largest number of uncovered nodes in $N_X$, and needs at most $O(n_1)$ time to include said nodes in $MTS(X,N_X)$, and needs at most $O(n_2)$ time to mark all nodes covered by nodes in $MTS(X,N_X)$ as covered. Hence, steps 1-3 take at most $O(n_1^2 + n_1 + 3n_2)$ time to complete. Since $n$ meets the following conditions: $n = n_1 + n_2$ and $n < |V|$, $O(n_1^2 + n_1 + 3n_2) < O(n^2 + n) < O(|V|^2 + |V|)$, which imply a time complexity of $O(|V|^2)$.

Similarly, the time complexity of MITS is bounded by step 2, where it needs to take at most $O(|V|^2)$ to find nodes with the maximum weight in the graph $G_Y$.\hfill\square

**Theorem 4.1.** The time complexity of CEN is $O(|V|^3)$. 

Proof. In total, when MTS and MITS are incorporated to reduce redundancy of trans-
migrations and avoid collision, it at most needs $|V|$ times for nodes in $V$ to operate
MTS and MITS. Hence the time complexity of CEN becomes $O(|V|^3)$. \hfill \square

The following lemmas concern the correctness of CEN.

**Lemma 4.2.** CEN provides a collision-free broadcast schedule.

Proof. This lemma is proven by contradiction. Assume there are two nodes $u_1$ and
$u_2$ with the same transmission time in CEN, and they share a common unvisited
neighbor, meaning a collision will occur when they transmit simultaneously. Ac-
cording to the rule of MITS, since node $u_1$ and $u_2$ are neighbors in graph $G_Y$, node
$u_1$ and $u_2$ must be assigned a different transmission time. Thus, the transmission
time of node $u_1$ and $u_2$ is different, which contradicts the assumption. \hfill \square

**Lemma 4.3.** CEN produces a broadcast schedule with 100% reachability.

Proof. The lemma is also proven by contradiction. Assume there exists a node $i$ in
the network which has not received a packet in CEN, but all of node $i$’s neighbors
are informed by CEN. Only two cases may lead to this state. First, consider the
case, where all of node $i$’s neighbors are excluded from transmission set by the MTS
function in CEN. According to MTS, if node $i$’s neighbor $j$ is excluded from the
MTS set, there should be an alternative neighbor $f$, instead of $j$, which covers node
$i$. Hence, this case is impossible and contradicts the operation of MTS. In the second
case, node $i$ is not informed by CEN because two or more of its neighbors transmit
simultaneously, causing a collision. According to MITS, if more than two senders
with a common neighbor, i.e., node $i$, transmit simultaneously, they will be scheduled
to transmit at a different time, as per Lemma 4.2. Therefore, the second case is also
contradictory to the function of MITS, which concludes the proof. \hfill \square

**Theorem 4.2.** CEN produces a correct broadcast schedule.

Proof. According to Lemma 4.2 and 4.3, each node in the network can receive the
broadcast packet collision-free. Therefore, CEN produces a correct and collision-free
broadcast schedule. \hfill \square

**Lemma 4.4.** Consider a node $i \in V$ that receives a broadcast packet at time slot
$rec(i)$ and transmits the packet at $tr(i)$. Let $\Delta$ denote the maximum degree in graph
$G$. Then, $tr(i) \leq rec(i) + (\Delta - 1)T$.

Proof. This lemma holds true for the source node $s$, as $tr(s) \leq rec(s)$. Next, it
proves this lemma also holds true for all other nodes in $V$. Node $i$ receives the packet
at time slot $rec(i)$. In the worst case, at every $T$ time slots, node $i$ will be considered
by MTS and MITS to assign it with a transmission time slot. Recall that node $i$ has
at most $\Delta$ one-hop neighbors. As a transmitting node, node $i$ must receive the packet from one of its neighbors and must transmit the packet to at least one of its neighbors that have not received the packet yet. Thus, node $i$’s transmission can be interfered by at most $(\Delta - 2)$ nodes. For instance, in the worst case, node $i$’s $(\Delta - 2)$ one-hop neighbors receive the packet at time slot $rec(i) + T, rec(i) + 2T, \ldots, rec(i) + (\Delta - 2)T$ and to achieve collision-free transmission, node $i$ will avoid transmitting at these time slots. Therefore, $tr(i) \leq rec(i) + (\Delta - 2)T + T = rec(i) + (\Delta - 1)T$.

**Lemma 4.5.** Consider nodes in layer $k$, where $0 \leq k \leq l$. Then their transmission time must be less than $t_0 + k(\Delta - 1)T$, where $t_0$ is the starting time slot.

**Proof.** This lemma is proven by induction. For layer 0, it is true, because source node $s$ transmits the packet at $t_0$. For nodes in layer 1, this lemma also holds true, because they receive the packet from source node $s$ at time slot $t_0$, and according to Lemma 4.4, their transmission time must be smaller than $t_0 + (\Delta - 1)T$, i.e., $tr(i) \leq t_0 + (\Delta - 1)T$, where $i \in L_1$. Assume this lemma holds true for all layers before $k$, and then prove it is also true for nodes in layer $k$. For nodes in layer $k$, they must receive the packet before $t_0 + (k - 1)(\Delta - 1)T$, because after time slot $t_0 + (k - 1)(\Delta - 1)T$, all layers before $k$ have already finished their transmissions, which cover all nodes in layer $k$. According to Lemma 4.4, the transmission time of nodes in layer $k$ must be less than $t_0 + (k - 1)(\Delta - 1)T + (\Delta - 1)T = t_0 + k(\Delta - 1)T$. Therefore, this lemma also holds true for layer $k$. 

**Theorem 4.3.** CEN gives a $(\Delta - 1)T$ approximate broadcast solution.

**Proof.** According to Lemma 4.5, the maximum transmission time in CEN is $t_0 + l(\Delta - 1)T$. Recall that $l \leq T(l)$ and $T(l)$ is the lower bound for the broadcast latency. It means the maximum transmission time in CEN must be less than $t_0 + T(l)(\Delta - 1)T$, and therefore, the broadcast latency for CEN is bounded by $t_0 + T(l)(\Delta - 1)T - t_0 = (\Delta - 1)TT(l)$. Hence, the approximation ratio of CEN is $(\Delta - 1)T$.

### 4.2.3 Distributed (DIS) Broadcast Algorithm

This section presents the distributed implementation of CEN. In DIS, it is assumed that each node has a unique ID and is aware of the ID of its two hops neighbors, which can be obtained from ‘HELLO’ messages sent by each node. The packet format and state information are first introduced before delving into the details. Lastly, a worked example is presented.
For each node $v \in V$, DIS focuses on finding the MTS and MITS set amongst its two hops neighbors. Node $v$ will apply the MTS and MITS function, described in Section 4.2.2, to assign a transmission sequence (broadcast time slot) to its one-hop neighbors, i.e., $N_1(v)$. This is achieved by eavesdropping on its neighbors’ broadcast in order to collect information in the packet header regarding the assigned transmission sequences. Node $v$ then either maintains the existing transmission sequences or creates a new transmission sequence for unscheduled one-hop neighbors.

Figure 4.5(a) shows the broadcast packet format to relay information required by nodes to compute a transmission sequence. It is comprised of two fields: Header and Data. The Header field includes the sender’s and its one-hop neighbors’ broadcast state information. The state information comprises of a node’s ID, RxBit, which is set to ‘1’ if said node has received the broadcast packet, RxTime, which records the reception time slot of a broadcast packet, TxBit, which is set to ‘1’ if the node has transmitted the broadcast packet, and TxTime is the broadcast packet’s transmission time slot. Lastly, the Data field denotes the payload. For instance, the Header of a broadcast packet sent by node $s$ in Figure 4.6 should contain the broadcast state information of node $s$, $v_1$, $v_6$ and $v_7$, e.g., for node $s$, RxBit=1, RxTime=1, TxBit=1 and TxTime=2.

![Figure 4.5](image)

**Figure 4.5**: An illustration of data format used in DIS

Additionally, node $v$ also maintains two data structures: MyStatus and MyNeigh-
bors. The former data structure, see Figure 4.5(b), records node $v$’s self-state information, i.e., ID, RxBit, RxTime, TxBit and TxTime, but additionally it includes RtxBit, which is set to ‘1’ if a node is required to retransmit the broadcast packet. Also, a CollisionBit is used to indicate whether the previous broadcast resulted in a collision; see Section 4.2.3.3. The MyNeighbors has the same data structure as Header in 4.5(a), but it is used to record all eavesdropped information from packet headers - i.e., information regarding one- and two-hop neighbors’ state information.

The following sections outline the distributed broadcast algorithm, and show how the information in each broadcast packet header is set.

### 4.2.3.1 Update State Information

A key problem in DIS to be addressed is inconsistent transmission sequences. Specifically, a node may be assigned a different transmission sequence by its neighbors. Consider a topology where node $v$ and $w$ are node $u$’s one-hop neighbors, but there is no link between node $v$ and $w$. Before node $u$’s transmission, node $v$ and $w$ have already transmitted a broadcast packet to $u$ at different time slots. Node $v$ and $w$ may assign a different transmission sequence to node $u$.

To address this problem, node $i$ needs to collect state information from its neighbors in order to update its MyStatus and MyNeighbors. This state information then allows node $i$ to learn the transmission sequences of its neighbors, i.e., $N_1(i) \cup N_2(i)$, or whether they have transmitted the broadcast packet. For neighbors in $N_1(i)$ with a transmission sequence, node $i$ avoids recalculating their transmission sequence. In addition, if a node receives different transmission sequences, it will only adopt the first transmission sequence it has received to avoid a potential conflict.

The following rules specify how node $i$ updates its MyStatus and MyNeighbors, which are represented as $M_1(i)$ and $M_2(i)$ respectively. Initially, $M_1(i)$ and $M_2(i)$ are empty, and RxBit, TxBit, RtxBit and CollisionBit are set to ‘0’, and both RxTime and TxTime are set to ‘$\infty$’.

The following rules apply when node $i$ receives a packet from node $j$. 
**R1:** If this is not the first instance of the broadcast packet for node $i$, no update is carried out to $M_1(i)$. Otherwise, it sets the RxBit to ‘1’, RxTime to be node $j$’s TxTime recorded in the packet Header, and node $i$’s TxTime is set to the corresponding TxTime specified by the node $j$ in the packet Header. Other information remains the same.

**R2:** As for $M_2(i)$, if RxBit of a node $v \in M_2(i)$ is ‘1’, node $i$ retains node $v$’s state information. Otherwise, node $i$ updates node $v$’s corresponding entry in $M_2(i)$ to the information specified in the received packet Header.

The following rules apply when node $i$ broadcasts a packet.

**R1:** For $M_1(i)$, node $i$ updates its TxBit to be ‘1’. If it does not need to retransmit the broadcast packet, it sets the RtxBit to be ‘1’; else, it is set to ‘0’.

**R2:** For $M_2(i)$, if RxBit of a node $v \in M_2(i)$’s is ‘1’, node $v$’s corresponding entry in MyNeighbors remains the same, else node $i$ updates node $v$’s RxBit to ‘1’, RxTime to be node $i$’s TxTime, TxBit to ‘0’ and TxTime to the scheduled transmission sequence calculated in Section 4.2.3.2.

**R3:** Generate a broadcast packet with its Header set to the state information in $M_1(i)$ and $M_2(i)$ and transmit it at the TxTime of node $i$.

### 4.2.3.2 Transmission Sequence Scheduling

Without loss of generality, assume node $i$ received a packet $p$ from its neighbor and needs to transmit the packet at time slot $t_i$. Based on its local state information in $M_1(i)$ and $M_2(i)$ at time $t_i$, node $i$ applies MTS and MITS function as per Section 4.2.2.1 and 4.2.2.2 to produce a broadcast schedule from its one-hop neighbors in $N_1(i)$ to its two-hop neighbors in $N_2(i)$. More specifically, node $i$ applies the following steps to determine its one-hop neighbors’ transmission sequence.

1. Include nodes in $N_1(i)$ and $N_2(i)$ into two temporary sets $A$ and $B$ respectively, i.e., $A = N_1(i)$ and $B = N_2(i)$. 
2. Remove nodes with TxBit of ‘1’ from $A$, and nodes with RxBit of ‘1’ from $B$. Nodes with TxBit=1 in $N_1(i)$ have transmitted the packet $p$, and nodes with RxBit=1 in $N_2(i)$ have received the packet $p$, and thus node $i$ does not need to schedule these nodes’ transmissions.

3. Divide nodes in $A$ into two subsets $A_1$ and $A_0$ respectively according to their RxBit value, i.e., $A_1 = \{j \mid j \in A \text{ and } \text{RxBit} = 1\}$ and $A_0 = \{j \mid j \in A \text{ and } \text{RxBit} = 0\}$.

4. For each node $i \in A_0$, assign a transmission sequence to it. Specifically, update node $i$’s TxTime to be $\min\{t \mid t > t_i \text{ and } t = \text{Tx}(i, k)\}$, where $k \in \mathbb{N}$.

5. Denote by $t_{\text{min}}$ the minimum TxTime value of nodes in $A_1 \cup A_0$. Include nodes in $A_1$ and $A_0$ with a TxTime of $t_{\text{min}}$ into the set $\overline{A}_1$ and $\overline{A}_0$ respectively. Note that, either $\overline{A}_1$ or $\overline{A}_0$ may be empty.

6. Set $X = \overline{A}_1 \cup \overline{A}_0$ and assign $N_X$ to be $X$’s one-hop neighbors that are in $B$. Build the sub-graph $G_X$ as per Section 4.2.2.1. Recall that, nodes in $\overline{A}_1$ have been assigned to a transmission sequence before, avoiding recalculating new transmission sequence, hence select nodes in $\overline{A}_1$ into MTS set $MTS(X, N_X)$ first and remove them and their adjacent nodes in $G_X$ from $G_X$ as well. Call MTS function to consider remaining nodes in $\overline{A}_0$ of $G_X$ into $MTS(X, N_X)$.

7. Set $Y = MTS(X, N_X)$. Build the traffic graph $G_Y$ as per Section 4.2.2.2. With the same reason of Step-6, select nodes in $\overline{A}_1$ into MITS set first, i.e., $MITS(Y, N_X)$, and remove them and their adjacent nodes in $G_Y$ from $G_Y$ as well. Call MITS function to consider remaining nodes in $G_Y$.

8. Remove nodes in $X \setminus MTS(X, N_X)$ from $A_0$ and update their transmission sequence, i.e., TxTime, to $\infty$, as they are redundant as per section 4.2.2.1. Also, remove nodes in $MITS(Y, N_X)$ from $A_1$ and $A_0$ respectively and set their transmission sequence to $t_{\text{min}}$, i.e., TxTime $= t_{\text{min}}$. Next, remove the neighbors of nodes in $MITS(Y, N_X)$ from $B$ as well. Recall that node $i \in Y \setminus MITS(Y, N_X)$ is delayed to transmit the broadcast packet as per Section 4.2.2.2, and thus its TxTime is set to $\min\{t \mid t > t_{\text{min}} \text{ and } t = \text{Tx}(i, k)\}$, where $k \in \mathbb{N}$. Repeat from Step-5 until $A_1 \cup A_0$ is empty.
4.2.3.3 Handling Collisions

Two or more nodes with a common neighbor may transmit a broadcast packet simultaneously with non-negligible probability, causing a collision. Fortunately, a receiving node is aware of the broadcast time slot of its neighbors; see Section 4.1. Upon detecting a corrupted broadcast message, the receiving node picks an interfering neighbor, say $v$, which has an overlapping broadcast time slot with other neighbors at random to carry out the broadcast again. The receiving node then sets the CollisionBit to ‘1’ in its MyStatus, and notifies node $v$ in its broadcast time slot. After receiving the notification message, the selected node $v$ updates its RtxBit to ‘1’ and resends the broadcast message at its broadcast time slot. If no message is received, the receiving node will repeat the above steps until it receives the broadcast message. Note that due to random broadcast time slots, a node will not experience persistent collisions or overlapping periods. Moreover, due to capture effect, a node may successfully receive from neighbor with the strongest signal. As a result, a node will finally get the broadcast packet successfully.

4.2.3.4 Example

The operation of DIS is illustrated using Figure 4.6. The network applies the same topology and configuration as the example of algorithm OPT. In this example, node $s$ is the source node, and it starts to transmit at time slot 2, i.e., TxTime=2. Initially, set $A = \{v_1, v_6, v_7\}$ and $B = \{v_2, v_3, v_5, v_8\}$. Recall that no nodes in $M_2(s)$ have transmitted or received the packet, i.e., RxBit=TxBit=0, and thus $A_1 = \{j \mid j \in A \text{ and } \text{RxBit} = 1\} = \{\}$ and $A_0 = \{v_1, v_6, v_7\}$. Update the value of TxTime for nodes in $A_0$ to 5, 5 and 5 respectively as per Table 4.2. Find the minimum TxTime among nodes in $A_1 \cup A_0$, i.e., $t_{\min} = 5$, and set $A_1 = \{\}$ and $A_0 = \{v_1, v_6, v_7\}$. Next, include nodes in $A_1 \cup A_0$ into set $X$, i.e., $X = \{v_1, v_6, v_7\}$, and their one-hop neighbors into $N_X$, i.e., $N_X = \{v_2, v_3, v_5, v_8\}$. Next, call MTS and MITS function to obtain $MTS(X, N_X) = \{v_1, v_7\}$ and $MITS(Y, N_X) = \{v_7\}$. Finally, as node $v_6$ is redundant, set its TxTime to $\infty$; node $v_1$ is not included in $MITS(Y, N_X)$, thus set its TxTime to 8, i.e., $\min\{t \mid t > 5 \text{ and } t = 8\}$ as per Table 4.2; update node $v_7$’s TxTime to $t_{\min} = 5$ because node $v_7$ is included in
Remove node $v_6$ and $v_7$ from $A_0$ as they have been scheduled with a transmission sequence, and remove their one-hop neighbors, i.e., node $v_3$, $v_5$ and $v_8$, from $B$. Then repeat the above steps with $A_1 = \{\}$ and $A_0 = \{v_1\}$ until $A_1 \cup A_0$ becomes empty. As a result, in $M_2(s)$, the TxTime of node $v_1$, $v_6$ and $v_7$ is set to 8, $\infty$ and 5 respectively. All other nodes are also handled in a similar manner and the final result is shown in Figure 4.6.

In this example, the broadcast latency of DIS is 9 and the number of transmissions is 6, i.e., node $s$, $v_1$, $v_3$, $v_5$ and $v_7$. Compared with the example in Section 4.2.2, DIS requires one more node, say $v_3$, to transmit the packet even though its one-hop neighbors $v_2$, $v_4$ and $v_7$ have already received the packet. This is because node $v_3$ determines its transmission based on its local state information. That is, when node $v_3$ transmits the packet at time slot 9, in its $M_2(v_3)$, node $v_2$ and $v_4$’s RxBit state is still ‘0’ even node $v_2$ and $v_4$ have already received the packet at time slot 8 and 6 respectively. It is because node $v_3$ has not received a packet from its one-hop neighbors which contains node $v_2$ and $v_4$’s latest state information after time slot 8.

![Figure 4.6: An illustration of DIS](image)

4.2.3.5 Discussion

Given that DIS needs to exchange state information on two hops basis, it is of interest to calculate the bounds of the broadcast packet header size, required memory, and the message complexity. The following lemmas assert the size of broadcast packet header used by DIS and the memory space required by each node to store the broadcast state information, and the upper bound on message complexity.
Lemma 4.6. The size of broadcast packet header used in DIS is upper-bounded by $O(\Delta + 1)$.

Proof. All packet headers in DIS contain the broadcast state information of the sender and its one-hop neighbors, where each node’s state information consists of an ID (two bytes), RxBit (one byte), RxTime (four bytes), TxBit (one byte) and TxTime (four bytes) fields, and has a constant size, i.e., $2 + 1 + 4 + 1 + 4 = 12$ bytes. Here, $\Delta$ denotes the maximum degree of graph $G$, and thus, the maximum size of a broadcast packet header is $12(\Delta + 1)$ bytes. Therefore, the size of a broadcast packet header is bounded by $O(\Delta + 1)$ in DIS.

Lemma 4.7. The size of broadcast state information stored in each node is bounded by a constant.

Proof. For each node, the broadcast state information used in DIS is stored in two data structures: MyStatus and MyNeighbors. For MyStatus, it stores local state information and has a constant size of 14 bytes, i.e., two bytes for ID, four bytes for RxBit, TxBit, RtxBit and CollisionBit, and eight bytes for RxTime and TxTime. For MyNeighbors, it contains the state information of neighbors among a node’s two hops. Each node has at most $\Delta^2$ one and two-hop neighbors, i.e., $\Delta$ one-hop neighbors and $(\Delta - 1)\Delta$ two-hop neighbors. Similar to Lemma 4.6, the maximum size of MyNeighbors is $12\Delta^2$ bytes. Therefore, the size of MyStatus and MyNeighbors is bounded by a constant.

Lemma 4.8. DIS has $O(|V|)$ message complexity.

Proof. The message complexity of DIS is $O(|V|)$ since each sender only sends out one broadcast packet. During the broadcast process of node $i \in |V|$, all node $i$’s one-hop neighbors wake up together to receive a broadcast packet from node $i$ as per the pseudo-random MAC. Therefore, the broadcast packet is only sent once by node $i$ to reach all of $i$’s one-hop neighbors, and thereby, yielding a total message complexity of $O(|V|)$.

4.3 Evaluation

This section outlines the research methodology used to evaluate the performance of CEN and DIS. It also compares them against OPT, flooding and OTAB [46]. Recall that OPT is used to obtain the theoretical broadcast latency bound. In the following experiments, each algorithm is measured against the following metrics:
• **Broadcast latency** – this is defined as the total time required by all nodes to receive a broadcast message;

• **Number of transmissions** – this is the total number of transmissions.

In these experiments, all nodes are stationary and randomly deployed in a square area of 1000 × 1000 m². An arbitrary node is selected as the source node. The effect of different network configurations is studied, including number of nodes, transmission radius and duty cycle, where the duty cycle is defined as the ratio of the duration of the active time slots to the total broadcast latency. The number of nodes ranges from 100 to 300. The transmission radius ranges from 50 to 250 meter. The duty cycle varies from 0.25 to 0.025. It is worth pointing out that experiments in Section 4.3 only aim to study the broadcast latency and number of transmissions caused by nodes’ varying wake-up times. Consequently, there is no need to employ a packet level simulator. Every experiment is conducted with one change to the network configuration while the other two are fixed. Every result is the average of 100 simulation runs, which corresponds to 10 random graphs and 10 random source nodes.

To date, there is no flooding algorithm especially designed for low duty-cycled networks; therefore, besides comparing CEN and DIS against OPT, this section also compared them against a modified flooding algorithm. To make the comparison as fair as possible, pure flooding is modified as follows, labelled *Improved Flooding*. First, it uses ArDeZ. Second, it uses the same method as DIS to handle collisions, see Section 4.2.3.3, when multiple senders are within communication range. As illustrated in Section 4.2.3.3, due to the pseudo-random property of ArDeZ and capture effect, a message has a high probability of being received successfully even in the presence of neighboring transmissions. These modifications thus improve pure flooding by reducing collisions and redundant transmissions significantly.

### 4.3.1 Results

#### 4.3.1.1 Number of Nodes

The performance of all algorithms is evaluated under different number of nodes. In this experiment, the transmission radius is fixed at 140m and the duty cycle is set to
0.05. In Figure 4.7, we can see that the broadcast latency of CEN and DIS decreases with increasing number of nodes. This is because as the number of nodes increases, the average degree rises correspondingly and for CEN and DIS, the sender can cover more neighbors via one transmission, resulting in lower broadcast latency. Moreover, the broadcast latency of CEN and DIS is much lower than that of Improved Flooding, i.e., the broadcast latency of DIS is only around 50% of that of improved flooding. Additionally, the broadcast latency of CEN and DIS is slightly higher than that of OPT, i.e., at most 53% higher.

![Figure 4.7: Broadcast latency under different number of nodes](image)

Figure 4.7 shows the impact on the number of transmissions when we increase the number of nodes. Observe that the transmission times of CEN and DIS are significantly lower than that of improved flooding, i.e., when the number of nodes grows to 300, the transmission times of DIS are only about 25% of that of improved flooding. In addition, the number of transmissions of all algorithms decreases as the number of nodes in the fixed area rises. This is because more nodes need to be covered.

### 4.3.1.2 Transmission Radius

Next, the performance of all algorithms is studied under different transmission radii. The number of nodes is set to 400 and the duty cycle of nodes is fixed at 0.05. As shown in Figure 4.9, the broadcast latency decreases when we increase the transmis-
Centralised and Distributed Algorithms under the RTS/CTS Interference Model

Figure 4.8: Number of transmissions under different number of nodes

sion radius. This is because nodes with a larger transmission radius are able to cover more nodes, which helps a sender to deliver the broadcast message to more nodes in less time. Moreover, CEN and DIS produce a much lower broadcast latency than improved flooding.

From Figure 4.10, we can see that the transmission times of DIS and CEN degrades as the transmission radius becomes larger. This is mainly because nodes are able to cover more nodes with a wider transmission radius, and it helps DIS and CEN to select fewer relaying nodes to cover the whole network. However, improved flooding needs more transmissions when the transmission radius increases because of more collisions resulting from a larger transmission range. Moreover, the number of transmissions of improved flooding is significantly greater than that of CEN and DIS, i.e., about three times greater on average.

4.3.1.3 Duty Cycle

Finally, the performance of all algorithms is studied under different duty cycles. The number of nodes is fixed to 100 and the transmission radius is set to 140m. From Figure 4.11, we can see that the broadcast latency of CEN and DIS decreases as nodes’ duty cycle grows. This is because when we increase a node’s duty cycle, it
Figure 4.9: Broadcast latency under different transmission radii

Figure 4.10: Number of transmissions under different transmission radii
will wake up more frequently leading to a reduction in the broadcast latency correspondingly. Moreover, the broadcast latency of improved flooding is around two times larger than that of all other algorithms. In addition, these curves also indicate CEN, DIS and OPT have a similar performance under different duty cycles.

![Broadcast Latency vs Duty Cycle](image)

**Figure 4.11:** Broadcast latency under different duty cycles

Figure 4.12 shows that with increasing duty cycle, the number of transmissions incurred by CEN and DIS decreases slowly. On the contrary, for improved flooding, it requires more transmissions to deliver a broadcast message to all nodes. With increasing duty cycles, more interfering nodes will select the same time slot as their broadcast time slot, leading to more collisions if these said senders transmit simultaneously. Thereby, senders need to retransmit a broadcast message multiple times. We see that the number of transmissions of improved flooding is almost two times larger than other algorithms.

### 4.3.1.4 Performance of CEN, DIS versus OTAB

In this section, CEN and DIS are compared against the state of the art algorithm for duty-cycled networks, i.e., OTAB. Recall that OTAB follows the conventional MAC model as in [38] [46], which requires a node to transmit a packet to a receiving node only when the receiving node is in its active time slot. In order to compare the proposed broadcast algorithms faithfully with OTAB, the MAC used by OTAB uses
the same duty cycle as ArDez MAC model, meaning nodes using OTAB have the same number of active time slots.

As shown in Figure 4.13(a), the broadcast latency of OTAB is around five times larger than that of proposed algorithms; i.e., CEN or DIS. This is mainly because OTAB is conducted layer by layer based on its shortest path tree $T_{SPT}$ and nodes in a lower layer are prevented from transmitting until all nodes in the current layer have finished their transmissions even though these transmissions do not cause any interference; instead, transmissions in CEN and DIS are handled in a greedy manner, which allows nodes to transmit as long as they do not result in collisions. Moreover, as compared with conventional MACs [38] [46], the inherent pseudo-random nature of ArDez also helps to reduce a large number of collisions, leading to a shorter broadcast latency. In addition, it is observed that the number of transmissions of CEN and DIS is only half that of OTAB. This is because CEN and DIS allow a node to transmit only one time to cover its one-hop neighbors as opposed to multiple times in OTAB.

4.3.1.5 Discussion

In summary, CEN and DIS achieve near optimal broadcast latency performance. From Section 4.3.1.1, 4.3.1.2 and 4.3.1.3, we can see that the broadcast latency of the two proposed algorithms is close to that of the optimal algorithm. In addition,
Figure 4.13: Performance of CEN, DIS versus OTAB
the two proposed algorithms perform much better than improved flooding and OTAB in terms of the broadcast latency and number of transmissions. The significant reduction in broadcast latency achieved by CEN and DIS is mainly due to nodes being scheduled in a greedy manner. The other reason is the pseudo-random MAC model used by CEN and DIS, which helps reduce the probability of overlapping wake-up times [38] [46]. Consequently, nodes experience less collision.

4.4 Conclusion

This chapter has presented an investigation into the MLBSDC problem under the RTS/CTS interference model. The main difficulty with this problem is that sensor nodes are not synchronized and do not wake up simultaneously. To overcome this problem, two novel algorithms, CEN and DIS, are designed for nodes that employ pseudo-random duty cycle schedules. The proposed algorithms apply two schemes, MTS and MITS, to reduce redundant transmissions and to assign transmission sequences. In particular, the MTS algorithm selects nodes that have good coverage, whilst MITS ensures transmitting nodes are scheduled at non-conflicting times. Hence, CEN and DIS are able to achieve collision-free broadcast and reduce the broadcast latency and redundancy efficiently. Finally, the performance of CEN and DIS are compared against OPT, improved flooding and OTAB. The simulation results show that both CEN and DIS have near optimal performance in terms of the broadcast latency and have lower redundancy of transmissions than improved flooding and OTAB under different network scenarios.

So far the MLBS problem has been studied under the RTS/CTS interference model. However, it is well known that a node’s interference range is much larger than its transmission range, and thus limits the number of transmitting and receiving nodes, which inevitably prolongs broadcast. To this end, the next chapter will study the MLBS problem under the protocol interference model, where the interference range is larger than the transmission range.
Chapter 5

Approximation Algorithms under the Protocol Interference Model

The interference resulting from a node’s transmission poses a key challenge to the design of any broadcast algorithms/protocols. In particular, it is well known that a node’s interference range is much larger than its transmission range and thus limits the number of transmitting and receiving nodes, which inevitably prolongs broadcast. To this end, a number of past studies have designed broadcast algorithms that account for this interference range with the goal of deriving a broadcast schedule that minimizes latency. However, these works have only taken into account interference that occurs within the transmission range of a sender, such as the algorithms proposed in Chapter 3 and 4. Therefore, the resulting latency is non-optimal given that collision occurs at the receiver. Henceforth, this chapter presents a study on the Interference-Aware Broadcast Scheduling (IABS) problem (also referred to as MLBS), which aims to find a schedule with the minimum broadcast latency subject to the constraint that a receiver is not within the interference range of any senders.

This chapter studies the IABS problem under the protocol interference model, and outlines two constant approximation algorithms, called Interference-Aware Basic Broadcast Scheduling (IABBS), and its enhanced version, Interference-Aware En-
hanced Broadcast Scheduling (IAEBS), which produce a latency of at most

\[
2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + \left(\frac{\pi}{2} + 1\right)(\rho + 1) + 1 \right\rfloor R \tag{5.1}
\]

where \( \rho \) is the ratio between the interference range and the transmission range, i.e., \( \rho \geq 1 \), and \( R \) is the radius of the network with respect to the source node of the broadcast. This chapter evaluated both algorithms under different network configurations and confirmed that the latencies achieved by IABBS and IAEBS are much lower than existing schemes. In particular, compared to CABS [55], the best constant approximation broadcast algorithm to date, the broadcast latency achieved by IAEBS is \( \frac{5}{8} \) of that of CABS. Moreover, it shows IABBS and IAEBS are also applicable for duty-cycled WSNs, which is the first attempt to address the MLBSDC problem under the protocol interference model.

## 5.1 Preliminaries

### 5.1.1 Network Model

It is assumed that all nodes have an equal transmission and interference range. Therefore, the network is represented by a UDG, \( G = (V, E) \), where \( V \) is the set of nodes, and \( E \) represents the set of edges/links that exist between two nodes if their Euclidean distance is no more than the transmission range. Denote the transmission range of nodes as \( r_T \), their interference range as \( r_I \), and \( \rho \geq 1 \) is the ratio between \( r_I \) and \( r_T \). \( N_1(v) \) and \( N_\rho(v) \) are used to denote the set of nodes that are within the transmission and interference range of node \( v \in V \) respectively, \( N_1(v) \subseteq N_\rho(v) \).

Time is assumed to be discrete and every message transmission occupies one unit time. In this chapter, the protocol interference model is adopted, which is widely used because of its generality and tractability [36]. In the protocol interference model, two simultaneous transmissions, i.e., ‘\( u_1 \to v_1 \)’ and ‘\( u_2 \to v_2 \)’, are said to be interference-free if none of a sender’s receivers are located within the other’s interference range; that is, \( d(u_1, v_2) > r_I \) and \( d(u_2, v_1) > r_I \), where \( d(u_1, v_2) \) (re-
spectively, \( d(u_2, v_1) \) is the Euclidean distance between \( u_1, v_2 \) (respectively, \( u_2, v_1 \)).

### 5.1.2 Graph Definitions and Theories

Let \( G = (V, E) \) be a connected and undirected UDG with \( |V| = n \), and node \( s \) is a fixed node in \( G \). The subgraph of \( G \) induced by \( U \subseteq V \) is denoted by \( G[U] \). The minimum degree of \( G \) is denoted by \( \delta(G) \). The inductivity of \( G \) is defined as \( \delta^*(G) = \max_{U \subseteq V} \delta(G[U]) \). The depth of a node \( v \in V \) is the hop distance between \( v \) and \( s \), and the radius of \( G \) with respect to \( s \), denoted by \( R \), is the maximum hop distance of all nodes from \( s \). The depth of a node \( v \) can be computed by constructing a BFS tree \( T_{BFS} \) from \( G \). For \( 0 \leq i \leq R \), the layer \( i \) of \( T_{BFS} \) consists of all nodes at depth \( i \), denoted by \( L_i \).

An **Independent Set** (IS) \( I \) in \( G(V, E) \) is defined as a subset of \( V \) such that \( u, v \in V, (u, v) \notin E \). A **Maximal Independent Set** (MIS) \( U \) is an independent set which is not a subset of any other independent sets. A subset \( U \) of \( V \) is a dominating set of \( G \) if each node not in \( U \) is adjacent to at least a member of \( U \). Clearly, every MIS of \( G \) is also a dominating set of \( G \). If set \( U \) is a dominating set of \( G \) and \( G[U] \) is connected, then \( U \) is called a **Connected Dominating Set** (CDS) of \( G \). It is known that the size of MIS does not exceed \( 4 \text{opt} + 1 \), where \( \text{opt} \) denotes the minimum size of a CDS of \( G \) [79].

A proper node **coloring** of \( G \) is an assignment of colors, labelled by natural numbers, to the nodes in \( V \) such that any pair of adjacent nodes receive different colors. Any node ordering \( v_1, v_2, \ldots, v_n \) of \( V \) induces a proper node coloring of \( G \) in the first-fit manner. Specifically, for \( i = 1 \) to \( n \), assign node \( v_i \) the least assigned color that is not used by any neighbor \( v_j \), where \( j < i \). A particular node ordering of interest is the **smallest-degree-last** ordering. For \( i = n \) to \( 1 \), it sets \( v_i \) to the node with the smallest degree in \( G[U] \), where \( U \subseteq V \) and initially \( U = V \). After that, \( v_i \) is removed from \( U \), and the process repeats until \( U \) is empty. It is well-known that the node coloring of \( G \) induced by a smallest-degree-last ordering uses at most \( 1 + \delta^*(G) \) colors [58].

**Theorem 5.1.** (Groemer Inequality [31]). Suppose that \( C \) is a compact convex set and \( U \) is a set of points with mutual distance at least one. Then,
\[ |U \cap C| \leq \frac{\text{area}(C)}{\sqrt{3}/2} + \frac{\text{peri}(C)}{2} + 1, \]

where \( \text{area}(C) \) and \( \text{peri}(C) \) are the area and perimeter of \( C \) respectively.

When the set \( C \) is a disk or a half-disk, it has the following corollary.

**Corollary 5.1.** Suppose that \( C \) (respectively, \( C' \)) is a disk (respectively, half-disk) of radius \( r \), and \( U \) is a set of points with mutual distances at least one. Then,

\[
|U \cap C| \leq \frac{2\pi}{\sqrt{3}} r^2 + \pi r + 1,
\]

\[
|U \cap C'| \leq \frac{\pi}{\sqrt{3}} r^2 + \left( \frac{\pi}{2} + 1 \right) r + 1,
\]

**Theorem 5.2.** (Mahjourian et al. [55]). In order for two simultaneous transmissions \('u_1 \rightarrow v_1' and 'u_2 \rightarrow v_2'\) to be interference-free according to the protocol interference model, it is sufficient to have,

\[
d(u_1, u_2) > (\rho + 1)r_T \lor d(v_1, v_2) > (\rho + 1)r_T
\]

where \( d(u_1, u_2) \) (respectively, \( d(v_1, v_2) \)) is the Euclidean distance between \( u_1, u_2 \) (respectively, \( v_1, v_2 \)).

A summary of notations used in this chapter can be found in Table 5.1.

### 5.1.3 Key Observation

Note that the conditions in Theorem 5.2 are the geometrical constraint used by [40], [55] and [75], and are in general stronger than what is needed for avoiding interfering transmissions. For example in Figure 5.1, assume that two transmissions \('u_1 \rightarrow v_1' and 'u_2 \rightarrow v_2'\) have the following geometrical property, \( d(u_1, u_2) < 3r_T \), \( d(v_1, v_2) < 3r_T \), \( d(u_1, v_2) > 2r_T \) and \( d(u_2, v_1) > 2r_T \) with \( \rho = 2 \). According to Theorem 5.2, \('u_1 \rightarrow v_1' and 'u_2 \rightarrow v_2'\) cannot be scheduled simultaneously by algorithms in [40] [55] and [75]. However, node \( v_1 \) and \( v_2 \) are outside the interference range of node \( u_1 \) and \( u_2 \) respectively, and hence transmissions \('u_1 \rightarrow v_1' and 'u_2 \rightarrow v_2'\) are interference-free. That is, they can be scheduled simultaneously.
Table 5.1
Definition of Notations in Chapter 5

<table>
<thead>
<tr>
<th>Notation</th>
<th>Definition</th>
<th>Notation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$G(V, E)$</td>
<td>Network graph</td>
<td>$U_i$</td>
<td>Nodes in $U \cap L_i$</td>
</tr>
<tr>
<td>$N_1(v)$</td>
<td>Nodes within $v$’s transmission range</td>
<td>$M_i$</td>
<td>Nodes in $M \cap L_i$</td>
</tr>
<tr>
<td>$N_\rho$</td>
<td>Nodes with $v$’s interference range</td>
<td>$\tau_I$</td>
<td>Transmission range</td>
</tr>
<tr>
<td>$T$</td>
<td>Scheduling period</td>
<td>$\tau_T$</td>
<td>Transmission range</td>
</tr>
<tr>
<td>$\tau(v)$</td>
<td>Transmission time of $v$</td>
<td>$C(v)$</td>
<td>$v$’s children</td>
</tr>
<tr>
<td>$\rho$</td>
<td>$\frac{r_I}{r_T}$</td>
<td>$P(v)$</td>
<td>$v$’s parent node</td>
</tr>
<tr>
<td>$T_{BFS}$</td>
<td>BFS tree</td>
<td>$L_i$</td>
<td>Nodes in layer $i$ of $T_{BFS}$</td>
</tr>
<tr>
<td>$l$</td>
<td>Maximum number of layers in $T_{BFS}$</td>
<td>$rec(v)$</td>
<td>Reception time of $v$</td>
</tr>
<tr>
<td>$T_b$</td>
<td>Broadcast tree</td>
<td>$color(v)$</td>
<td>$v$’s color</td>
</tr>
<tr>
<td>$G_t$</td>
<td>Transmitting conflict graph</td>
<td>$G_r$</td>
<td>Receiving conflict graph</td>
</tr>
<tr>
<td>$m_t$</td>
<td>Chromatic number in $G_t$</td>
<td>$m_r$</td>
<td>Chromatic number in $G_r$</td>
</tr>
<tr>
<td>$\delta(G)$</td>
<td>Minimum degree in $G$</td>
<td>$T_{start}$</td>
<td>Current time</td>
</tr>
<tr>
<td>$G[X]$</td>
<td>Subgraph of $G$ induced by $X \subseteq V$</td>
<td>$\delta^*$</td>
<td>$\max_{X \subseteq V} \delta(G[X])$</td>
</tr>
<tr>
<td>$R$</td>
<td>Radius of $G$ with respect to $s$</td>
<td>$n$</td>
<td>Number of nodes in $G$</td>
</tr>
</tbody>
</table>

5.2 Proposed Algorithms

The ensuing sections first present IABBS followed by its enhanced version, IAEBS, which has a near optimal performance.

![Figure 5.1: An example of two simultaneous transmissions](image-url)
5.2.1 IABBS

The main idea of IABBS is to schedule transmissions layer by layer based on the rule that a transmission is interference-free if there are no other senders within a receiving node’s interference range. IABBS starts by constructing a broadcast tree $T_b$ rooted at the source node $s$, where if a node $u$ is a parent of node $v$, then node $u$ is responsible for transmitting the message to $v$. Then using $T_b$, IABBS schedules transmissions layer by layer such that every node receives the message interference-free. In the following explanation, Figure 5.2 and Figure 5.3 are used as an example to illustrate key aspects of IABBS. The network in Figure 5.2 consists of 13 nodes randomly deployed in a $4 \times 5$ rectangle area, and node $s$ is the source node.

![Figure 5.2: An example wireless network with $r_T = 1$ and $r_I = 2$. There are 13 nodes randomly deployed in a $4 \times 5$ rectangle area. Solid lines denote transmission range. Dotted lines denote the interference range.](image)

Firstly, IABBS constructs a BFS tree $T_{BFS}$ rooted at node $s$, and computes the depth of all nodes in the resulting $T_{BFS}$. Hence, this tree yields the radius $R$ of $G$; for the topology in Figure 5.2, it has $R = 5$, see Figure 5.3. With this tree in hand, let $L_i$, where $0 \leq i \leq R$, be the set of nodes at depth $i$ of $T_{BFS}$.

The next step is to construct the broadcast tree $T_b$. For example, the resulting $T_b$ for
the network depicted in Figure 5.2 is shown in Figure 5.3. This tree will be used to determine the transmitting nodes and their transmission schedule. The construction of $T_b$ has two key features: (i) deriving a MIS $U$, and (i) selecting nodes, called *dominators*, from the set $U$, and their parents, also called *connector* nodes, such that dominators together with connectors form a CDS.

Algorithm 5.1 constructs the MIS $U$ layer by layer, starting from $L_0$ in $T_{BFS}$ (line 4 to 9). Specifically, for each layer $L_i$, it selects nodes that are not adjacent to nodes in $U$ greedily. Let $U_i = U \cap L_i$, and $M_i = L_i \setminus U_i$; in Figure 5.3, $U_2 = \{v_1, v_2, v_5\}$ and $M_3 = \{v_{10}, v_{11}\}$. Note that, $U_0 = \{s\}$ and $U_1 = \emptyset$ because the source node $s$ is the first node to be considered and nodes in $L_1$ must be adjacent to node $s$.

Given the sets $U_i$ and $M_i$, the next step is to select parent nodes. At each layer $i$, where $0 < i \leq R$, Algorithm 5.1 greedily selects parents from $U_i$ that cover the most nodes in the current and lower layers that have yet to be assigned a parent; see line 14 to 18. Let $P(v)$ and $C(v)$ be the parent of node $v$, and the set of children of node $v$ respectively. For instance, in Figure 5.3, for nodes in layer 4, initially, node $v_8 \in U_4$ covers the most nodes in $M_4 \cup M_5$, i.e., $M_4 = \emptyset$ and $M_5 = \{v_6, v_{12}\}$, so it is first chosen as the parent of node $v_6$ and $v_{12}$, i.e., $P(v_6) = P(v_{12}) = v_8$ and $C(v_8) = \{v_6, v_{12}\}$. To identify the connector nodes, lines 19 to 23 in Algorithm 5.1 process nodes in a similar manner; i.e., it selects as connectors nodes in $M_i$ that cover the most dominators in the lower layer that have yet to be assigned a parent, whereby nodes in $M$ serve as parents to nodes in $U$.

After constructing $T_b$, the next step is to schedule nodes’ transmissions using Algorithm 5.4. For each layer of $T_b$, dominators first transmit followed by connectors. IABBS schedules transmissions with the help of two conflict graphs $G_t$ and $G_r$, where an edge between two nodes indicate interference and hence must not transmit or receive simultaneously. These two conflict graphs are constructed based on the rule that an edge exists between two transmitting nodes in $G_t$ (respectively, two receiving nodes in $G_r$) if any of their children (respectively, parents) lie in the interference range of one another. The following paragraphs explain these conflict graphs in more detail.
Algorithm 5.1 Broadcast Tree $T_b$

1: $T_b \leftarrow (V_b, E_b)$, $V_b \leftarrow V$, $E_b \leftarrow \emptyset$
2: $U \leftarrow U_0 \leftarrow \{s\}$
3: $P(v) \leftarrow \text{Nil}, \forall v \in V$
4: for $i \leftarrow 1$ to $R$
5: for each $w \in L_i$
6: if ($U \cap N_1(w)) = \emptyset$
7: $U_i \leftarrow U_i \cup \{w\}; U \leftarrow U \cup \{w\}$
8: end if
9: end for
10: $M_i = L_i \setminus U_i$
11: end for
12: for $i \leftarrow 0$ to $R$
13: $U'_i \leftarrow U_i$, $M'_i \leftarrow M_i$
14: while $U'_i \neq \emptyset$
15: $u \leftarrow$ node in $U'_i$ with maximum $|\{v \mid v \in N_1(u) \cap L_{j>i} \text{ and } P(v) = \text{NIL}\}|$ value
16: $C(u) \leftarrow \{v \mid v \in N_1(u) \cap L_{j>i} \text{ and } P(v) = \text{NIL}\}$
17: $U'_i \leftarrow U'_i \setminus \{u\}$ and $P(v) \leftarrow u$, $\forall v \in C(u)$
18: end while
19: while $M'_i \neq \emptyset$
20: $u \leftarrow$ node in $M'_i$ with maximum $|\{v \mid v \in N_1(u) \cap U_{j>i} \text{ and } P(v) = \text{NIL}\}|$ value
21: $C(u) \leftarrow \{v \mid v \in N_1(u) \cap U_{j>i} \text{ and } P(v) = \text{NIL}\}$
22: $M'_i \leftarrow M'_i \setminus \{u\}$ and $P(v) \leftarrow u$, $\forall v \in C(u)$
23: end while
24: end for
25: $E_b \leftarrow \{(u, v) \mid u = P(v)\}$
26: return $T_b = (V_b, E_b)$
Figure 5.3: Broadcast tree $T_b$ of IABBS. The label $(rec, tr)$ denotes the reception and transmission time of a node, and a square indicates a dominator.

Graph $G_t$ is constructed as per Algorithm 5.2 which is used to ensure all dominators’ transmissions in $U_i$ are interference-free. Algorithm 5.2 constructs graph $G_t$ by taking as input $U_i$ and $T_b$, and outputs $G_t(V_t, E_t)$. First, all nodes in $U_i$ that have children in layer $i$ or $i + 1$ are added into $V_t$ (line 3 to 5 in Algorithm 5.2). Next, Algorithm 5.2 will connect two nodes in $V_t$ with an edge if they have children that lie in the interference range of one another (line 8 to 14 in Algorithm 5.2). This means two sending or dominator nodes that are connected by an edge in $G_t$ must not be scheduled to transmit simultaneously because a subset of a dominator’s children lies within the interference range of the other.

IABBS constructs the conflict graph $G_r$ using Algorithm 5.3. $G_r$ is then used to ensure that the reception of dominators in $U_{i+1}$ is interference-free. In other words, IABBS ensures the transmissions of connectors in $M_i$ are interference-free because the parent of nodes in $U_{i+1}$ are connectors in $M_i$. Algorithm 5.3 takes as input of $U_{i+1}$ and $T_b$, and outputs a subgraph $G_r(V_r, E_r)$. Specifically, all nodes in $U_{i+1}$ are added into $V_r$ (line 2 in Algorithm 5.3), and then two nodes in $V_r$ are connected with an edge if they do not have as their parent the same connector, and at least one receiving node is interfered by the other’s parent node, hence neither of them can be scheduled to receive at the same time.
Algorithm 5.2 Conflict Graph $G_t(V_t, E_t)$
1: Procedure $\text{Conflict-Graph-}G_t(U_i, T_b)$
2: $V_t \leftarrow E_t \leftarrow \emptyset$
3: \textbf{for} each $u \in U_i$ \textbf{do}
4: \hspace{1em} \textbf{if} $C(u) \neq \emptyset$ \textbf{then}
5: \hspace{2em} $V_t \leftarrow V_t \cup \{u\}$
6: \hspace{1em} \textbf{end if}
7: \textbf{end for}
8: \textbf{for} each $u \in V_t$ \textbf{do}
9: \hspace{1em} \textbf{for} each $v \in V_t$ \textbf{do}
10: \hspace{2em} \textbf{if} $C(u) \cap N_\rho(v) \neq \emptyset$ \textbf{then}
11: \hspace{3em} $E_t \leftarrow E_t \cup (u,v)$
12: \hspace{2em} \textbf{end if}
13: \hspace{1em} \textbf{end for}
14: \textbf{end for}
15: \textbf{return} $G_t = (V_t, E_t)$

Algorithm 5.3 Conflict Graph $G_r(V_r, E_r)$
1: Procedure $\text{Conflict-Graph-}G_r(U_{i+1}, T_b)$
2: $V_r \leftarrow U_{i+1}$, $E_r \leftarrow \emptyset$
3: \textbf{for} each $u \in V_r$ \textbf{do}
4: \hspace{1em} \textbf{for} each $v \in V_r$ \textbf{do}
5: \hspace{2em} \textbf{if} $P(u) \neq P(v)$ and $P(u) \cap N_\rho(v) \neq \emptyset$ \textbf{then}
6: \hspace{3em} $E_r \leftarrow E_r \cup (u,v)$
7: \hspace{2em} \textbf{end if}
8: \hspace{1em} \textbf{end for}
9: \textbf{end for}
10: \textbf{return} $G_r = (V_r, E_r)$
After constructing the conflict graphs of layer $i$, IABBS proceeds to color the nodes in $G_t$ and $G_r$, where nodes in $G_t$ (respectively, $G_r$) that share the same color are scheduled to transmit (respectively, receive) at the same time. To minimize their chromatic index, IABBS takes advantage of the smallest-degree-last ordering method to color nodes in the first-fit manner; see Algorithm 5.4 (line 6 and 15). Denote by $\text{color}(v)$ the color number of node $v$, i.e., $\text{color}(v) = 0, 1, 2, \ldots$. Let $m_t$ and $m_r$ be the maximum number of colors required by nodes in graph $G_t$ and $G_r$ respectively.

**Algorithm 5.4** Broadcast Scheduling of IABBS

1: $tr(v) \leftarrow \text{rec}(v) \leftarrow -1, \forall v \in V$
2: $T_{\text{start}} \leftarrow 0$
3: for $i \leftarrow 0$ to $R$ do
   4:     if $U_i \neq \emptyset$ then
   5:         $G_t(V_t, E_t) \leftarrow \text{Conflict-Graph-}G_t(U_i, T_b)$
   6:         Color nodes in $G_t$ by smallest-degree-last ordering
   7:         for each $v \in V_t$ do
   8:             $tr(v) \leftarrow T_{\text{start}} + \text{color}(v)$
   9:             $\text{rec}(u) \leftarrow tr(v), \forall u \in C(v)$
 10:         end for
 11:         $T_{\text{start}} \leftarrow T_{\text{start}} + m_t$
 12:     end if
 13:     if $U_{i+1} \neq \emptyset$ then
 14:         $G_r(V_r, E_r) \leftarrow \text{Conflict-Graph-}G_r(U_{i+1}, T_b)$
 15:         Color nodes in $G_r$ by smallest-degree-last ordering
 16:         for each $v \in V_r$ do
 17:             $tr(P(v)) \leftarrow T_{\text{start}} + \text{color}(v)$
 18:             $\text{rec}(v) \leftarrow tr(P(v))$
 19:         end for
 20:         $T_{\text{start}} \leftarrow T_{\text{start}} + m_r$
 21:     end if
22: end for
23: return $\text{rec}(v), \forall v \in V$

IABBS schedules the transmissions from parents at layer $i$ to their children as follows. Specifically, the transmission of a dominator $u$ in $U_i$ to its children $C(u)$ is
scheduled at time $T_{\text{start}} + \text{color}(u)$ based on graph $G_t(V_t, E_t)$, where $T_{\text{start}}$ is the current time (line 7 to 10 of Algorithm 5.4). The current time $T_{\text{start}}$ increases by $m_t$ to ensure all transmissions from dominators in $U_i$ completes (line 11 in Algorithm 5.4). Then, transmissions from connectors in $M_i$ to dominators in $U_{i+1}$ are scheduled in a similar manner. Note that, $V_r = U_{i+1}$. The reception of dominator $u$ in $U_{i+1}$ is also scheduled at time $T_{\text{start}} + \text{color}(u)$ based on graph $G_r(V_r, E_r)$, and accordingly the transmission time $tr(P(u))$ of node $u$’s parent $P(u)$ is set to node $u$’s reception time $\text{rec}(u)$, i.e., $tr(P(u)) = \text{rec}(u) = T_{\text{start}} + \text{color}(u)$ (line 16 to 19 in Algorithm 5.4). The current time $T_{\text{start}}$ increases by $m_r$ (lines 20 in Algorithm 5.4) so that all transmissions from connectors in $M_i$ finishes before the next layer is considered. All subsequent layers are then scheduled in a similar manner.

Figure 5.3 is now used as the example to illustrate the operation of Algorithm 5.4. It starts by constructing graph $G_t$ and $G_r$ for layer 0. Recall that $U_0 = \{s\}$, and $U_1 = \emptyset$, hence graph $G_r$ is skipped. Graph $G_t$ only contains one node $s$, therefore, $\text{color}(s) = 0$ and $m_t = 1$. Then, the transmission time $tr(s)$ of node $s$ is set to $T_{\text{start}} + \text{color}(s) = 0$, where initially $T_{\text{start}} = 0$. Next, the current time $T_{\text{start}}$ increases by $1$, $T_{\text{start}} = 1$. For layer 1, $U_1 = \emptyset$, and thus graph $G_t$ is empty. It only needs to construct graph $G_r$ from layer 1 with $U_2 = \{v_1, v_2, v_3\}$. Recall that node $v_1$ and $v_5$ share the same parent $v_7$, i.e., $P(v_1) = P(v_5) = v_7$, both node $v_1$ and $v_5$ lie in the interference range of node $v_3$ which is the parent of node $v_2$, and node $v_2$ is within the interference range of node $v_7$, subsequently, in $G_r$, there is a link between node $v_1$ and $v_2$, $v_5$ and $v_2$ respectively, i.e., $(v_1, v_2) = (v_5, v_2) = 1$. Then sort the nodes in $G_r$ as per smallest-degree-ordering to yield a new scheduling order $\{v_2, v_1, v_5\}$. Next, color these nodes in the first-fit manner in the following order: $\{v_2, v_1, v_5\}$. Then $\text{color}(v_2) = 0$, $\text{color}(v_1) = \text{color}(v_5) = 1$ and $m_r = 2$. Finally, the the reception time of node $v_2$, $v_1$ and $v_5$ is set to 1,2 and 2 respectively, i.e., $\text{rec}(v_2) = T_{\text{start}} + \text{color}(v_2) = 1 + 0$. Correspondingly, the transmission time of node $v_3$ and $v_7$ is also set to 1 and 2, i.e., $tr(v_3) = \text{rec}(v_2) = 1$ and $tr(v_7) = \text{rec}(v_1) = \text{rec}(v_2) = 2$. The other layers are handled in a similar manner and the final result is shown in Figure 5.3.
5.2.2 IAEBS

In this section, an enhancement, IAEBS, is presented to IABBS. IAEBS differs from IABBS only in terms of how transmissions are scheduled. Instead of scheduling transmissions layer-by-layer in a top-down manner, IAEBS is able to schedule transmissions across multiple layers. This means in IAEBS, nodes in lower layers may receive or transmit the broadcast message earlier than nodes in an upper layer. As shown in Figure 5.4, this helps reduce the broadcast latency.

Similar to IABBS, IAEBS starts by constructing the broadcast tree $T_b$ using Algorithm 5.4. After that, for each $L_i$, it uses Algorithm 5.2 and 5.3 to construct $G_t$ and $G_r$. Instead of coloring nodes in $G_t$ and $G_r$ as IABBS, IAEBS first sorts nodes in $G_t$ and $G_r$ according to the smallest-degree-last ordering, and records them in a new set $Q_t$ and $Q_r$ respectively. Then, it schedules the transmissions of nodes in $L_i$ based on $G_t$ and $G_r$ greedily.

More specifically, any transmitting node $u$ in $G_t$ transmits at the minimum time $t$ that satisfies the following interference-free constraints – (i) $u$ must receive the message interference-free before time $t$, i.e., $tr(u) > rec(u)$ (line 10 of Algorithm 5.5); (ii) no node in $C(u)$ hears the message from nodes in its interference range at time $t$ (line 7 in Algorithm 5.5); (iii) no node in $N_\rho(u) \setminus C(u)$ is receiving a message from its parent at time $t$ (line 8 in Algorithm 5.5). Likewise, any receiving node $v$ in $G_r$ receives at the minimum time $t$ that satisfies similar constraints – (i) the reception time $t$ of node $v$ must be larger than $rec(P(v))$ (line 21 in Algorithm 5.5); (ii) node $v$ is not hearing a message from nodes in its interference range at time $t$ (line 18 in Algorithm 5.5); (iii) no node in $N_\rho(P(v)) \setminus C(P(v))$ is receiving a message from its parent at time $t$ (line 19 in Algorithm 5.5). In Algorithm 5.5, set $I_1(v)$ and $I_2(v)$ are used to record the time in constraint (ii) and (iii) for node $v$ respectively. Denote by $I(v)$ the set $I_1(v) \cup I_2(v)$. Note that, nodes in $G_t$ are scheduled before nodes in $G_r$ because all parents of nodes in $G_r$ are assigned a reception time only when nodes in $G_t$ are scheduled to transmit.

Figure 5.4 is used as an example to illustrate the operation of Algorithm 5.5. IAEBS will construct the same broadcast tree $T_b$, conflict graph $G_t$ and $G_r$ for each layer $i$, ...
Algorithm 5.5 Broadcast Scheduling of IAEBS

1: $tr(v) \leftarrow rec(v) \leftarrow -1, \forall v \in V$
2: for $i \leftarrow 0$ to $R$ do
3:   if $U_i \neq \emptyset$ then
4:     $G_t(V_t, E_t) \leftarrow \text{Conflict-Graph-} G_t(U_i, T_b)$
5:     Sort nodes in $G_t$ by smallest-degree-last ordering and
6:     Use $Q_t$ to denote nodes in $V_t$ with the new order
7:     for each $u \in Q_t$ do
8:       $I_1(u) \leftarrow \{t \mid \exists w \in C(u) \text{ that hears a message at } t \text{ from } N_{\rho}(w)\}$
9:       $I_2(u) \leftarrow \{t \mid \exists w \in N_{\rho}(u) \setminus C(u) \text{ that is scheduled to receive at } t\}$
10:      $I(u) \leftarrow I_1(u) \cup I_2(u)$
11:      $tr(u) \leftarrow \min\{t \mid t > rec(u) \text{ and } t \notin I(u)\}$
12:      $rec(w) \leftarrow tr(u), \forall w \in C(u)$
13:     end for
14:   end if
15: if $U_{i+1} \neq \emptyset$ then
16:     $G_r(V_r, E_r) \leftarrow \text{Conflict-Graph-} G_r(U_{i+1}, T_b)$
17:     Sort nodes in $G_r$ by smallest-degree-last ordering and
18:     Use $Q_r$ to denote nodes in $V_r$ with the new order
19:     for each $v \in Q_r$ do
20:       $I_1(v) \leftarrow \{t \mid \exists v \text{ that hears a message at } t \text{ from } N_{\rho}(v) \setminus \{P(v)\}\}$
21:       $I_2(v) \leftarrow \{t \mid \exists w \in N_{\rho}(P(v)) \setminus C(P(v)) \text{ scheduled to receive at } t\}$
22:       $I(v) \leftarrow I_1(v) \cup I_2(v)$
23:       $tr(P(v)) \leftarrow \min\{t \mid t > rec(P(v)) \text{ and } t \notin I(v)\}$
24:       $rec(v) \leftarrow tr(P(v))$
25:     end for
26: end if
27: end for
28: return $rec(v), \forall v \in V$
and the final broadcast tree $T_b$ is shown in Figure 5.4. In the next step, IAEBS will schedule transmissions for each layer. It starts by sorting the nodes in $G_t$ and $G_r$ for layer 0. $U_1 = \emptyset$, hence $G_r$ for layer 0 is empty. It has $tr(s) = 0$, i.e., $I_1 = I_2 = \emptyset$, $rec(v_3) = rec(v_7) = 0$ because $G_t$ only contains node $s$, $Q_t = \{s\}$. For layer 1, since $U_1$ is empty, IAEBS only needs to consider nodes in $U_2$. Hence, it will sort the nodes in $G_r$ as per the smallest-last-degree ordering, and yields $Q_r = \{v_2, v_1, v_5\}$ for layer 1. Node $v_2$ is first considered in $Q_r$. As node $s$ is the only transmission at time 0, $I_1(v_2) = \{0\}$, $I_2(v_2) = \{0\}$ and $rec(v_3) = 0$. Thus, $tr(v_3) = rec(v_2) = 1$.

Next, node $v_1$ is considered. Node $v_1$ hears a message from $s$ and $v_3$ at time 0 and 1 respectively, therefore $I_1(v_1) = \{0, 1\}$. For node $v_7$ as node $v_1$’s parent, among its interference range, node $v_3$ and $v_2$ are scheduled to receive the broadcast message at time 0 and 1 respectively; That is, $I_2(v_1) = \{0, 1\}$. Thus, $tr(v_7) = rec(v_1) = 2$, as $rec(v_7) = 0$ and $I(v_1) = \{0, 1\}$. Node $v_5$ is the last node to be scheduled. Since $rec(v_7) = 0$, $I_1(v_5) = \{0, 1\}$ and $I_2(v_5) = \{0, 1\}$, $tr(v_7) = rec(v_5) = 2$, i.e., $\min\{t \mid t > 0 \text{ and } t \notin \{0, 1\}\} = 2$. The other layers are handled in a similar manner, and the final result is shown in Figure 5.4. Note that, the reception time of node $v_{11}$ in layer $L_3$ is equal to that of node $v_1$ and $v_5$ in layer 2.
5.2.3 Analysis

The following sets of theorems assert the correctness, time complexity and approximation ratio of IABBS and IAEBS in terms of the broadcast latency and number of transmissions.

**Theorem 5.3.** IABBS yields a correct and interference-free broadcast schedule.

*Proof.* Recall that IABBS processes nodes’ transmissions layer by layer in a top-down manner, and the transmissions at each layer are only scheduled after all those in upper layers have completed. Thus it only needs to prove all nodes in each layer can be scheduled interference-free. That is, for each layer, nodes in the conflict graph $G_t$ and $G_r$ are interference-free when they are scheduled to transmit or receive. The correctness for each layer is proven by contradiction. Assume node $u$ and $v$ in $G_t$ transmit at the same time $t$ to their respective children. Assume node $u$’s children are in the interference range of node $v$. This means there is a link between node $u$ and $v$ in $G_t$, according to Algorithm 5.2. Thus, node $u$ and $v$ must not share the same color. That is, node $u$ and $v$ must not be scheduled to transmit simultaneously. This is contradictory to the assumption. So for each layer, nodes in $G_t$ transmit interference-free. Similarly, it can prove all nodes in $G_r$ receive the broadcast message interference-free. Consequently, this theorem holds true.

**Theorem 5.4.** IAEBS yields a correct and interference-free broadcast schedule.

*Proof.* The correctness of this theorem is proven by contradiction. It is assumed that node $v$ cannot be scheduled to receive interference-free because there are two or more parallel transmissions to node $v$ at the same time. Assume that node $v$’s parent $P(v)$ and one of the nodes in $N_p(v)$, i.e., $u$, are scheduled to transmit at $t$. Furthermore, consider two different cases. In the first case, node $P(v)$ is scheduled before node $u$. If node $P(v)$ selects time $t$ as $P(v)$’s transmission time, node $u$ will not choose $t$ again because by the third constraint (line 8 and 19 in Algorithm 5.5), when node $v$’s reception time is set to $t$, i.e., $rec(v) = tr(P(v)) = t$, node $u$ must not choose $t$. In the second case, assume node $u$ is scheduled before node $P(v)$. According to the second constraint (lines 7 and 18 in Algorithm 5.5), after node $u$ selects time $t$ as its transmission time, node $P(v)$ will not choose it again, because node $v$ will hear node $u$’s transmission at time $t$, i.e., $v \in N_p(u)$. This is contradictory to the assumption, so this theorem is true.

**Theorem 5.5.** IABBS produces a constant approximate solution for the IABS problem with latency at most $2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor R$. 


Proof. Observe that for each layer $i$, where $0 \leq i \leq R$, it takes $m_t + m_r$ unit time to finish all transmissions, thus we only need to prove the maximum value of $m_t$ and $m_r$ to obtain the maximum latency for IABBS. Recall that $m_t$ and $m_r$ are defined as the maximum number of colors required by dominators in $G_t$ and $G_r$ respectively. IABBS applies the smallest-degree-last ordering to color nodes in $G_t$ and $G_r$ respectively, hence $m_t = \delta^*(G_t) + 1$ and $m_r = \delta^*(G_r) + 1$ by [58]. In the worst case, Algorithm 5.2 and 5.3 will add a link between any two dominators whose Euclidean distance is no larger than $(\rho + 1)r_T$ by Theorem 5.2. The maximum minimum degree of any node $u$ in $G_t$ or $G_r$ is bounded by the number of nodes which lie in a half-disk of radius $(\rho + 1)r_T$ centred at $u$. All nodes in $G_t$ and $G_r$ are dominators. Therefore, by Theorem 5.1, the maximum minimum degree of node $u$ is bounded by $\left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor - 1$. That is, $\delta^*(G_t)$, $\delta^*(G_r) \leq \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor$. As a result, each layer will take at most $m_t + m_r = \delta^*(G_t) + \delta^*(G_r) + 2 \leq 2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor$ unit time to finish all transmissions. Hence, the maximum broadcast latency of IABBS is $2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor R$.

\[ \square \]

Theorem 5.6. IAEBS yields a constant approximate solution for the IABS problem with latency at most $2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor R$.

Proof. Recall that the transmission schedule of nodes is derived in a top-down manner greedily. Assume the maximum transmission time of nodes in layer $i$, where $0 \leq i < R$, is $T_i$. Suppose that node $u$ in $L_{i+1}$ and $G_t$ is scheduled to transmit after $T_i$. It only needs to consider nodes in $G_t$ which have been scheduled before node $u$ because all nodes in layer $i$ finish their transmissions after $T_i$. The scheduling order of nodes in $G_t$ is determined as per smallest-last-order, and thus when node $u$ is considered, at most $\delta^*(G_t)$ nodes have been considered before it. After $T_i$, at most $\delta^*(G_t)$ nodes will interfere with node $u$’s transmission, i.e., $|I(u)| \leq \delta^*(G_t)$. Consequently, the maximum transmission time of nodes in $G_t$ for layer $i + 1$ is $T_i + \delta^*(G_t) + 1$.

Next, nodes in $G_r$ are scheduled after nodes in $G_t$ according to IAEBS. Suppose that node $v$ in $G_r$ is scheduled to receive the broadcast message after $T_i + \delta^*(G_t) + \delta^*(G_r) + 2$, and only transmissions from the parents of nodes in $G_r$ interfere with node $v$’s reception. Similar to nodes in $G_t$, the scheduling order of nodes in $G_r$ is also determined by smallest-last-order, and thus when node $v$ is considered, at most $\delta^*(G_r)$ nodes have been scheduled to receive. Hence, the maximum reception time of nodes in $G_r$ for layer $i + 1$ is $T_i + \delta^*(G_t) + \delta^*(G_r) + 1$, and the maximum transmission time of parents of nodes in $G_r$ is $T_i + \delta^*(G_t) + \delta^*(G_r) + 2$. We get the maximum transmission time $T_{i+1}$ of nodes in layer $i + 1$ is $T_i + \delta^*(G_t) + \delta^*(G_r) + 2$.

By theorem 5.5, $\delta^*(G_t)$, $\delta^*(G_r) \leq \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + (\frac{\pi}{2} + 1)(\rho + 1) + 1 \right\rfloor - 1$, it gets
for each layer $i+1$, where $0 \leq i < R$, its maximum transmission time $T_{i+1}$ is bounded by $T_i + 2 \left[ \frac{\pi}{\sqrt{3}}(\rho+1)^2 + \left( \frac{\pi}{2} + 1 \right)(\rho+1) + 1 \right]$.

Thus it gets the maximum transmission time $T_i$ for each layer $i$ is bounded by $2 \left[ \frac{\pi}{\sqrt{3}}(\rho+1)^2 + \left( \frac{\pi}{2} + 1 \right)(\rho+1) + 1 \right] i$, where $0 \leq i \leq R$. Hence, the maximum latency yielded by IAEBS is $2 \left[ \frac{\pi}{\sqrt{3}}(\rho+1)^2 + \left( \frac{\pi}{2} + 1 \right)(\rho+1) + 1 \right] R$.

\[\square\]

**Theorem 5.7.** IABBS and IAEBS are 8-approximate solutions in terms of the number of transmissions.

**Proof.** Recall that IABBS and IAEBS use the same method, i.e., Algorithm 5.1, to construct the broadcast tree $T_b$. In $T_b$, only dominators and connectors are allowed to transmit a message. Each dominator transmits at most once and a connector may transmit several times to inform all of its dominator children. Given that each connector is a parent node of dominators in $U$, the number of transmissions by all connectors is equal to the number of dominators in $U$ except the source node $s$ which does not have a parent. The number of dominators is $|U|$, and thus, the total number of transmissions of dominators and connectors is $|U| + |U| - 1 = 2|U| - 1$. Recall that the size of $U$ does not exceed $4 \cdot \text{opt} + 1$ [79], where opt is the minimum number of transmissions. IABBS and IAEBS are thus a $2(4 \cdot \text{opt} + 1) - 1 = 8 \cdot \text{opt} + 1$ solution.

It is known that for a UDG, a node can be adjacent to at most five dominators [24]. Therefore, each connector is adjacent to at most five dominators in $U$, and one of these dominator is assigned as its parent. A connector may transmit at most four times, because for any connector it has at most four children in $U$.

\[\square\]

**Theorem 5.8.** The time complexity of IABBS and IAEBS is $O(n^2)$.

**Proof.** In the following, it first shows the broadcast tree (Algorithm 5.1) and conflict graphs (Algorithm 5.2 and 5.3) can be determined in time $O(n^2)$, then proves the broadcast scheduling (Algorithm 5.4 and 5.5) of IABBS and IAEBS can also be finished in time $O(n^2)$.

First, the BFS tree can be constructed in time $O(n^2)$. Then, IABBS and IAEBS need to iterate through $|E|$ edges to find MIS. So it will take at most $O(|E|) \leq O(n^2)$ time. To construct a conflict graph, each vertex will traverse at most $n$ nodes to build a link. To sum up, it takes at most time $\sum O(n_i \times n) \leq O(n^2)$, where $n_i$ is the number of vertices in each conflict graph. Above all, these steps will be finished in time $O(n^2)$.

The common step in broadcast scheduling of IABBS and IAEBS is the smallest-degree-last ordering for each conflict graph which takes at most time $O(n_i^2)$ in [58].
To sum up all conflict graphs, this step requires time \( \sum O(n_i^2) \leq O(n^2) \). Then, for IABBS, it needs to iterate at most \(|E|\) times to assign transmitting time; in other words, this step is determined in time \( O(|E|) \leq O(n^2) \). For IAEBS, each vertex in conflict graphs will iterate at most \( n \) times to determine the minimum transmission time. For all conflict graphs, it takes at most time \( \sum O(n_i \times n) \leq O(n^2) \).

Hence the total time complexity of IABBS and IAEBS is \( O(n^2) \).

\[\square\]

### 5.3 Evaluation

This section first outlines the research methodology used to evaluate the performance of IABBS and IAEBS. In particular, their performance versus BFS and CABS [55], which are known to have the best performance to date under the protocol interference model. Note that BFS outputs the depth of the BFS tree and can be used to obtain the lower broadcast latency bound, assuming no interference. In particular, for CABS, \( r_{S}/r_{T} \) is set to 0, where \( r_{S} \) is the carrier sensing range; i.e., the carrier sensing range is not considered in simulations. It is worth pointing out that the main goal of the evaluation is to compare the theoretical and experimental broadcast latency performance of IABBS and IAEBS. To this end, the focus is on the effect of various network configurations, explained below, on the broadcast latency and number of transmissions.

In each experiment, all nodes are stationary and randomly deployed in a \( 700 \times 700 \) m\(^2\) square area. The effect of different network configurations is studied, including number of nodes and transmission radius. The number of nodes ranges from 100 to 300. The transmission radius ranges from 70 to 160 meters. Every experiment is conducted with one change to the network configuration whilst the others are fixed. Each experiment is conducted on 20 randomly generated topologies. Moreover, for each topology, the experiment is carried out for 10 runs, and in each run, an arbitrary node is selected as the source node. Each result is the average of 200 simulation runs. The simulations are performed in MATLAB.
5.3.1 Results

5.3.1.1 Number of Nodes

Figure 5.5 is a plot of broadcast latency versus the number of nodes. Broadcast latency is the maximum time taken by any node to receive the message. We can see from the figure that broadcast latency does not vary very much with the number of nodes. This is because broadcast latency is mostly influenced by the depth of the BFS tree, which does not depend on the number of nodes. As shown in Figure 5.5, the depth of BFS tree does not fluctuate significantly. Moreover, IABBS and IAEBS have better performance than CABS, i.e., the broadcast latency produced by CABS is about 40 time units; in contrast, IABBS and IAEBS perform much better with a broadcast latency of 30 and 25 respectively. This is because instead of adopting Theorem 5.2 to schedule nodes’ transmissions, IABBS and IAEBS schedule two parallel transmissions if the corresponding children do not lie in one another’s interference range. This means two senders or receivers with Euclidean distance less than $(\rho + 1)r_T$ but satisfying the condition that their children or parents are not within the interference range of one another can be scheduled to transmit or receive simultaneously, and thereby, leading to a lower latency than CABS. Additionally, IAEBS performs better than IABBS because IAEBS schedules transmissions in more than one layer; that is, nodes in a lower layer may transmit or receive earlier.
From Figure 5.6, we observe that the transmission ratio for all algorithms decreases when the number of nodes increases. Here, the transmission ratio is the ratio between the number of transmissions of a broadcast message and number of nodes in the network. This is mainly because the average degree grows with increasing the number of nodes in a fixed network field, thereby, allowing one transmission to reach more nodes; That is, on average, each node needs fewer transmissions to cover its neighbors. Moreover, IABBS and IAEBS perform better than CABS in terms of transmission ratio.

![Graph](image1)

**Figure 5.6**: Transmission ratio under different number of nodes

### 5.3.1.2 Transmission Radius

![Graph](image2)

**Figure 5.7**: Broadcast latency under different transmission radii
Figure 5.7 is a plot of the broadcast latency versus the transmission radii. It shows that the broadcast latency of all algorithms decreases with increasing transmission radius. It is because as the transmission radius increases, the number of nodes being covered by each transmission also increases, leading to a reduction in the depth of the BFS tree, which is the key factor that influences the performance of CABS, IABBS and IAEBS. As shown in Figure 5.7, the depth of the BFS tree reduces with increasing transmission radius. Furthermore, IABBS and IAEBS perform better than CABS.

Figure 5.8 shows that the transmission ratio for IABBS, IAEBS and CABS also decreases as the transmission radius grows. When the transmission radius increases, a transmitting node can cover more nodes through one transmission, and thus fewer nodes will be needed to retransmit a broadcast message. This leads to a decline in transmission ratio. Furthermore, IABBS and IAEBS still keep a better performance with respect to transmission ratio than CABS.

![Transmission Ratio vs Transmission Radius](image)

**Figure 5.8:** Transmission ratio under different transmission radii

### 5.4 Remarks on Duty-Cycled WSNs

IABBS and IAEBS are also applicable in asynchronous, duty-cycled WSNs, where nodes determine their wake-up schedule independently and randomly. As a result, a node needs to transmit a broadcast message multiple times because only a subset
of its neighbors may be awake at any given point in time. Consider the same duty-
cycle aware MAC model defined in Chapter 3, in which if a node wants to transmit
a message, it will wake up at the corresponding receiver’s wake-up time slot to send
the message. IABBS and IAEBS address the MLBS problem in duty-cycled WSNs
as follows. They first construct a broadcast tree $T_b$ rooted at the source node as per
Algorithm 5.1, then use Algorithm 5.2 and 5.3 to construct conflict graphs $G_t$ and $G_r$
for each layer in $T_b$. After that, Algorithm 5.4 and 5.5 need to be adapted as follows
to support duty-cycled case.

When IABBS is applied in duty-cycled WSNs, each parent node needs to transmit at
the active time slots of its own children. Therefore, in line 8 of Algorithm 5.4, a node
$v$ in graph $G_t$ needs to transmit at $T_{start} + color(v)T + \tau(u)$, in which $u \in C(v)$. Here, $T$
is the scheduling period for a given duty-cycled WSN, and $\tau(u)$ is node $u$’s active time slot. Similarly, for each node $v$ in graph $G_r$, its parent node, i.e., $P(v)$, will transmit at $T_{start} + color(v)T + \tau(v)$. That is, line 17 of Algorithm 5.4 needs to be $tr(P(v)) \leftarrow T_{start} + color(v)T + \tau(v)$. Accordingly, line 11 and 20 of Algorithm 5.4 should increase $T_{start}$ by $m_tT$ and $m_rT$ respectively after each iteration. Note that, a parent node may be assigned with multiple transmission times based on its children’s active time slots. For each layer of $T_b$, IABBS takes at most $(m_t+m_r)T$ time slots. Hence, according to Theorem 5.3, IABBS in duty-cycled case will produce a solution with a ratio of $2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + \left(\frac{\pi}{2} + 1\right)(\rho + 1) + 1 \right\rfloor T$ for the broadcast latency. Furthermore, for duty-cycled WSNs, IABBS is a $8T$-approximate solutions in terms of the number of transmissions, because the neighbors of a node have at most $T$ different active time slots.

Similar to IABBS, IAEBS also only allows each parent to transmit at its child’s active
time slot. To be specific, in line 10 of Algorithm 5.5, each node $w$ in $G_t$ needs to transmit at $\min\{t \mid t \mod T = \tau(w), t > \text{rec}(u) \text{ and } t \notin I\}$, where $w \in C(u)$. Likewise, in line 21 of Algorithm 5.5, for each node $v$ in $G_r$, its parent node $P(v)$ needs to transmit at $\min\{t \mid t \mod T = \tau(v), t > \text{rec}(v) \text{ and } t \notin I\}$. Similarly, IAEBS also gives a solution with a ratio of $2 \left\lfloor \frac{\pi}{\sqrt{3}}(\rho + 1)^2 + \left(\frac{\pi}{2} + 1\right)(\rho + 1) + 1 \right\rfloor T$ for the broadcast latency, and $8T$ for the number of transmissions respectively.
5.5 Conclusion

This chapter has studied the interference aware broadcast scheduling problem in always-on wireless networks and duty-cycled WSNs. To overcome this problem, it designed two novel algorithms, IABBS and IAEBS, for nodes that use the protocol interference model. It proves that both algorithms provide correct and interference-free schedule, and produce a low broadcast latency. Furthermore, IABBS and IAEBS achieve a constant approximation ratio of \( 2 \left\lfloor \frac{\pi}{\sqrt{3}} (\rho + 1)^2 + \left( \frac{\pi}{2} + 1 \right) (\rho + 1) + 1 \right\rfloor \) in terms of the broadcast latency. Extensive simulation results show that both algorithms have better performance in terms of the broadcast latency and number of transmissions as compared to CABS under different network scenarios.

Chapter 3, 4 and 5 have studied the MLBS problem under the bounded interference models, i.e., the RTS/CTS and protocol interference model. However, these models are overly idealistic in that they do not model interference resulting from many far-away nodes, which could still have non-negligible effect on reception. To overcome this drawback, the next chapter will study the MLBS problem under the physical interference model, where the cumulative interference of many nodes outside the interference range is not neglected.
Distributed Algorithm under the Physical Interference Model

Thus far, existing studies such as those in Chapter 3, 4 and 5 address the MLBS problem over highly theoretical disk graph models, in which the transmission and interference range is thought of as a disk centred at a node. Specifically, these works deal with interference through the RTS/CTS model, see Chapter 3 and 4, or the protocol interference model, see Chapter 5. Nodes that adopt such interference models assume there is interference when nodes lie in the overlapping region within their interference range. These ‘interfered nodes’ must therefore be scheduled in different time slots. The main drawback of these interference models is that they cannot model the case where many far-away nodes could still have non-negligible effect on reception. To this end, the physical interference model, also called SINR-based interference model, is more realistic, where the cumulative interference of many nodes outside the interference range are not neglected.

Henceforth, this chapter considers MLBS for duty-cycled wireless networks over the physical interference model and presents the first distributed approximation algorithm called Hexagon Broadcast Algorithm (HBA). Despite being a greedy heuristic method, which considers the set of all nodes holding a broadcast message at any point in time as potential transmitters, the analysis in Section 6.3 shows that HBA
has a constant approximation ratio in terms of broadcast latency; specifically,

\[ 9 \left[ 2 + \frac{2}{3} \left( \frac{8\beta}{2} \left( \frac{2}{\alpha - 2} + \frac{1}{\alpha - 1} + 3 \right) \right)^{1/\alpha} \right]^2 T \]  

(6.1)

where \( \alpha \) is the path-loss exponent, \( \beta \) is the minimum SINR threshold required for a message to be decoded successfully, \( r_{\text{max}} \) is the maximum transmission range, \( r \) is nodes’ transmission range, and \( T \) is the scheduling period. The total number of transmissions in terms of broadcast messages produced by HBA is upper-bounded by \( (T + 1)N_H \), where \( N_H \) is the number of hexagons required to cover the entire network. HBA is evaluated under different network configurations and the results show that the latencies achieved by HBA are much lower than existing schemes. In particular, compared to the Tree-based Algorithm [40], the broadcast latency achieved by HBA is about \( \frac{1}{2} \) of that of Tree-based Algorithm.

### 6.1 Preliminaries

#### 6.1.1 Network Model

Consider nodes placed on an Euclidean plane, and \( d(u, v) \) represent the Euclidean distance between node \( u \) and \( v \). In addition, these nodes have uniform power level assignment, whereby all senders transmit with the same power level \( P \). As mentioned earlier, this chapter adopts the physical interference model, which is also called SINR-based interference model, where a receiver \( v \) successfully receives a message from a sender \( u \) if and only if the following condition holds:

\[ \frac{P \ d(u, v)^{-\alpha}}{\sum_{w \in V \setminus \{u, v\}} P \ d(w, v)^{-\alpha} + N} \geq \beta \]  

(6.2)

where \( V \) denotes the set of nodes in the network, \( \alpha \) is the path-loss exponent that is normally between 2 and 6, \( \beta \) denotes the minimum SINR required for a message to be received successfully which is greater than one, \( N \) is the ambient noise and \( \sum_{w \in V \setminus \{u, v\}} P \ d(w, v)^{-\alpha} \) is the interference experienced by node \( v \) from nearby nodes.

A scheduling period has \( T \) slots of fixed, equal length. Each slot is indexed by \( 1, 2, 3, \ldots, T \). Each time slot is assumed to be of sufficient duration to transmit or
receive a broadcast message. The network is locally synchronized at the slot level. As shown in [33], this can be achieved by local synchronization techniques, such as FTSP [56], which can yield an accuracy of 2.24 µs using only a few small packet exchanges among neighboring nodes every 15 minutes. It is important to note that this accuracy is sufficient as the active duration of each node is typically above 10,000 µs [20] [32]. Moreover, transmissions are not required to start at the beginning of each slot, meaning nodes do not need strict synchronization in order to communicate.

Given the scheduling period, the duty cycle is thus defined as the ratio between active time and $T$. For example, if $T = 10$, a 10% duty cycle means nodes are only awake in one slot. Considering the same duty-cycle aware MAC model defined in Chapter 3, each node $v$ selects one active time slot from $[1, 2, 3 \ldots, T]$ randomly and independently, and wakes up at its chosen time slot to receive a message. If node $v$ wants to transmit a message, it can wake-up to send this message at any time slot as long as the receiver is awake.

Finally, it is assumed that each node is aware of its location. This can be achieved by using localization methods such as [62] and [61] or nodes may be equipped with GPS. They also know the location of the base station, which is located at position $(0, 0)$. A summary of notations used in this chapter can be found in Table 6.1.

<table>
<thead>
<tr>
<th>$G_T$</th>
<th>Transmission graph</th>
<th>$\alpha$</th>
<th>Path loss exponent</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>SINR threshold</td>
<td>$P$</td>
<td>Transmission power</td>
</tr>
<tr>
<td>$N$</td>
<td>Ambient noise</td>
<td>$N_1(v)$</td>
<td>$v$'s one-hop neighbors in $G_T$</td>
</tr>
<tr>
<td>$T$</td>
<td>Scheduling period</td>
<td>$N_2(v)$</td>
<td>$v$'s two-hop neighbors in $G_T$</td>
</tr>
<tr>
<td>$r_{max}$</td>
<td>Maximum transmission range</td>
<td>$r$</td>
<td>Transmission range in $G_T$</td>
</tr>
<tr>
<td>$d(u,v)$</td>
<td>Distance between $u$ and $v$</td>
<td>$T_b$</td>
<td>Broadcast structure</td>
</tr>
<tr>
<td>$\mathcal{H}$</td>
<td>A hexagon</td>
<td>$color$</td>
<td>Node’s color label</td>
</tr>
<tr>
<td>$\mathcal{F}_i^1$, $\mathcal{F}_i^2$, $\mathcal{F}_i^3$</td>
<td>Sub-frame of $\mathcal{F}_i$</td>
<td>$\mathcal{F}_i$</td>
<td>$i$-th frame</td>
</tr>
<tr>
<td>$A\mathcal{F}$, $B\mathcal{F}$</td>
<td>Sub-part of $\mathcal{F}_i^1(t)$</td>
<td>$c$</td>
<td>Source node $s$’s color</td>
</tr>
<tr>
<td>$R$</td>
<td>Radius of $G$ with respect to $s$</td>
<td>$Lat$</td>
<td>Broadcast latency</td>
</tr>
<tr>
<td>$\mathcal{H}_1(v)$</td>
<td>Nodes in the same hexagon with $v$</td>
<td>$\mathcal{H}_2(v)$</td>
<td>One-hop neighbors of $\mathcal{H}_1(v)$ but in different hexagons from $v$</td>
</tr>
<tr>
<td>$S_r(t)$</td>
<td>Receptors in the same hexagon with the same active time $t$</td>
<td>$S_p(t)$</td>
<td>Providers of receptors in $S_r(t)$ not yet to send a message</td>
</tr>
</tbody>
</table>
6.1.2 Definitions and Theories

A transmission graph with respect to a range $r$ is defined as a connected graph $G_T = (V_T, E_T(r))$, where $E_T(r) = \{(u, v) \mid d(u, v) \leq r\}$. Let $r_{\text{max}} = \left(\frac{P}{N\beta}\right)^{1/\alpha}$, which is the maximum transmission range in the absence of interference from other simultaneous transmissions. Let $r_{\text{min}}$ be the length of the longest edge in the minimum spanning tree of $G_T$. In other words, $r_{\text{max}}$ and $r_{\text{min}}$ are the maximum and minimum $r$ such that the transmission graph $G_T$ remains connected; i.e., $r_{\text{min}} \leq r \leq r_{\text{max}}$. Let $N_1(u)$ denote the set of one-hop neighbors of node $u$, i.e., $N_1(u) = \{v \mid d(u, v) \leq r\}$. Accordingly, for a set $V$ of nodes, $N_1(V)$ denotes the set of one-hop neighbors of nodes in $V$, i.e., $N_1(V) = \bigcup_{u \in V} N_1(u)$. $N_2(u)$ denotes the two-hop neighbors of $u$, where node $v \in N_2(u)$ should share at least one common one-hop neighbor with $u$ and $r < d(u, v) \leq 2r$.

In a distributed environment, it is assumed each node knows the identity (ID), position and active time slot of its two-hop neighbors. This information can be gathered readily from any local broadcast techniques, e.g., [29] [37] or [91]. Incidentally, these techniques are the first to achieve local broadcast under the SINR-based interference model. Note, in practice, the required information can be embedded in ‘HELLO’ messages sent out by nodes during neighbor discovery.

A link $(u, v)$ is defined as the transmission from sender $u$ to receiver $v$, where $(u, v) \in E_T(r)$. Let $L$ denote a set of links in $G_T$. The links in set $L$ are said to be independent if all senders in $L$ can transmit simultaneously, and their corresponding receiver is able to decode each transmission successfully. The next theorem gives the sufficient condition for $L$ to be independent, and its proof can be found in the Appendix A.5.

**Theorem 6.1.** In order for set $L$ to be independent, it is sufficient for one of the following to be true:

1. The mutual distance of senders are all greater than $\rho r$;
2. The mutual distance of receivers are all greater than $\rho r$.

where $\rho = 1 + \left(\frac{8\beta}{1-(r/r_{\text{max}})^\alpha} \left(\frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3\right)\right)^{1/\alpha}$.

In practice, $\rho$ is a small constant. Consider $\alpha = 4$ and $\beta = 1$. Figure 6.1 indicates...
the relationship between $r/r_{max}$ and $\rho$. We see that when $r/r_{max} \leq 0.8$, $\rho$ is smaller than 4.

![Graph showing the relationship between $r/r_{max}$ and $\rho$.](image)

**Figure 6.1**: A plot of $\rho$ when $\alpha = 4$ and $\beta = 1$

![Hexagonal tessellation and coloring](image)

**Figure 6.2**: An example hexagonal tessellation and coloring

*Tessellation* is a way to partition a plane into equal (or similar) pieces. In Figure 6.2 we see a regular hexagonal tessellation. Given this tessellation, hexagons can be colored using a number of methods; examples include [39] [40] and [75]. Without loss of generality, the proposed algorithm to be described later will employ the following $3k^2$-coloring method when scheduling broadcast. As we will see in Section 6.2.2,
Algorithm 6.1 3$k^2$-coloring Method

1: **input:** $r, k, \vec{x}$ and $\vec{y}$
2: **output:** 3$k^2$-colored hexagons
3: $color \leftarrow 1$
4: **for** $i \leftarrow 1$ to $2k$ **do**
5:   **if** $i \leq k$ **then**
6:     **for** $j \leftarrow 1$ to $k + i$ **do**
7:       $\vec{h} \leftarrow ((2j - i - 1)\sqrt{3}r/4, -(i - 1)3r/4)$
8:     **end for**
9:   **else**
10:     **for** $j \leftarrow 1$ to $3k - i$ **do**
11:       $\vec{h} \leftarrow ((2j + i - 1 - 2k)\sqrt{3}r/4, -(i - 1)3r/4)$
12:     **end for**
13: **end if**
14: **Assign** $color$ **to all hexagons with centre $\vec{h} + ak\vec{x} + bk\vec{y}$, $\forall a, b \in \mathbb{Z}$
15: $color \leftarrow color + 1$
16: **end for**

the color of a hexagon will be used to achieve interference-free transmissions – that
is, nodes located in hexagons with a different color are not allowed to transmit or
receive simultaneously.

Given a natural number $k$ and a hexagonal tessellation with a hexagon radius of $r/2$,
let $r$ denote the transmission range in $G_T$. Define two vectors $\vec{x} = (3\sqrt{3}r/4, 3r/4)$
and $\vec{y} = (3\sqrt{3}r/4, -3r/4)$. These vectors have a length of $3r/2$. Repeat the follow-
ing process, see Algorithm 6.1, to assign a $color$ to all hexagons; here, $color$ is an
integer in the range $[1, 3k^2]$, i.e., $1 \leq color \leq 3k^2$. Start from an uncolored hexagon
with centre $\vec{h}$, and then assign all hexagons with centre at $\vec{h} + ak\vec{x} + bk\vec{y}$ with $color$,
where $a, b \in \mathbb{Z}$. For instance, give $k = 2$. Algorithm 6.1 needs to assign the same
$color$ value to all hexagons with center located at $\vec{h} + 3a\vec{x} + 3b\vec{y}$ with $a, b \in \mathbb{Z}$. This
process repeats until $color = 3k^2 = 12$. The result of 12-coloring is shown in Figure
6.2. Note, $\vec{h} + ak\vec{x} + bk\vec{y}$ is a function of $a$ and $b$; both of which can be arbitrary
integers, e.g., $a = 1, b = -1$, if there exists a hexagon in the network with centre at
$\vec{h} + ak\vec{x} + bk\vec{y}$.
Lemma 6.1. (Huang et al. [40]). Algorithm 6.1 results in a $3k^2$-coloring, and hexagons of the same color are separated by at least a distance of $(3k - 2)r/2$.

According to Theorem 6.1 and Lemma 6.1, in order to apply Algorithm 6.1 under the SINR-based interference model, we need to set $(3k - 2)r/2 = \rho r$. In other words, $k = \lceil 2(\rho + 2)/3 \rceil$. Based on the transmission graph $G_T$, a hexagon is said to be covered by a node $v \in V_T$, if and only if the said hexagon does not include $v$, but contains a subset of $v$’s one-hop neighbors. To distinguish nodes on the edges of hexagons, each hexagon is assumed to be left half open and right half close, meaning the top most node will be included whilst the bottom most node is excluded from the hexagon.

6.2 Distributed Broadcast Schedule

This section presents a distributed Hexagon Broadcast Scheduling Algorithm (HBA). It first describes HBA followed by the theoretical analysis confirming this algorithm has $O(1)$-approximation ratio in terms of broadcast latency.

6.2.1 Broadcast Structure

HBA starts by constructing a broadcast structure, denoted by $T_b$, before using the color of hexagons to derive a broadcast schedule such that nodes located in hexagons with a different color are not allowed to transmit or receive simultaneously.

Firstly, this section describes the construction of $T_b$; see Algorithm 6.2 for details. Each node first tessellates the network into equal hexagons with a radius of $r/2$ and then gives a $3([2(\rho + 2)/3])^2$-coloring to all hexagons (line 5 and 6 in Algorithm 6.2). Note, as the radius of each hexagon is $r/2$, the maximum distance in each hexagon is $r$; that is, nodes located in the same hexagon are one-hop neighbors of each other.

Next, based on two-hop neighbors information, node $v$ places into set $\mathcal{H}_1(v)$ neighbors that are in the same hexagon as itself, and adds into set $\mathcal{H}_2(v)$ nodes in the other
hexagons that are the one-hop neighbors of nodes in $\mathcal{H}_1(v)$ (line 7 and 8 in Algorithm 6.2). Note that set $\mathcal{H}_1(v)$ includes node $v$. Since nodes in $\mathcal{H}_1(v)$ are one-hop neighbors of each other and they are aware of two-hop neighbors information, nodes in the same hexagon will produce the same $\mathcal{H}_1$ and $\mathcal{H}_2$ sets.

To reduce the number of transmissions, HBA selects a set of nodes from $\mathcal{H}_1(v)$ that covers all neighboring hexagons containing a neighbor; i.e., the selected nodes are neighbors of nodes in $\mathcal{H}_2(v)$. Ideally, the chosen set should have a small cardinality. Specifically, HBA applies line 9 to 18 in Algorithm 6.2 to produce the broadcast structure $T_b = (V_b, E_b)$, where $V_b$ contains nodes used to relay broadcast messages, i.e., providers and receptors, and $E_b$ indicates the set of links between a provider and its corresponding receptor. Here, provider is a node selected from $\mathcal{H}_1(v)$ and is used for relaying a broadcast message to its corresponding receptors; while a receptor is a node chosen from set $\mathcal{H}_2(v)$ and is used to transmit a broadcast message to all other nodes in its hexagon.

Initially, HBA marks all hexagons as uncovered, and then repeats the following iterations until $\mathcal{H}_2(v)$ is empty. It first picks a node $u \in \mathcal{H}_1(v)$ that covers the most uncovered hexagons (line 11 in Algorithm 6.2), and labels $u$ as a provider. The next step is to select one corresponding receptor of $u$ from each uncovered hexagon. Specifically, for each uncovered hexagon, HBA will choose as the corresponding receptor a node $w$ with the smallest ID among nodes in $N_1(u)$; see line 13 in Algorithm 6.2. Then, it includes link $(u, w)$ in the set $E_b$, and removes nodes in $\mathcal{H}_1(w)$ from $\mathcal{H}_2(v)$, i.e., $\mathcal{H}_2(v) \setminus \{\mathcal{H}_1(w) \cap \mathcal{H}_2(v)\}$ (line 14 and 15 in Algorithm 6.2). It then marks the uncovered hexagon as covered. Note, provider $u$ and its corresponding receptor $w$ are located in different hexagons, and provider $u$ (respectively, receptor $w$) has only one corresponding receptor $w$ (respectively, provider $u$) in the hexagon including $w$ (respectively, $u$).

After the execution of Algorithm 6.2, the broadcast structure $T_b$ is constructed, where $V_b$ contains providers and receptors, and $E_b$ indicates the link of a provider and its corresponding receptors.

To illustrate the operation of Algorithm 6.2, consider Figure 6.3. Note, this exam-
Algorithm 6.2 Broadcast Structure $T_b$

1: **input:** Transmission graph $G_T = (V_T, E_T(r))$
2: **output:** $T_b = (V_b, E_b)$
3: $V_b \leftarrow \emptyset$, $E_b \leftarrow \emptyset$
4: for each node $v$ in $V_T$ do
5:   Tessellate the plane into equal hexagons with radius $r/2$, one of which is centred at $(0, 0)$
6:   Apply Algorithm 6.1 to color all hexagons by setting $k = \lceil 2(\rho + 2)/3 \rceil$
7:   $\mathcal{H}_1(v) \leftarrow \{u \mid u \text{ lies in the same hexagon as } v\} \cup \{v\}$
8:   $\mathcal{H}_2(v) \leftarrow \{u \mid u \in N_1(\mathcal{H}_1(v)) \text{ and } u \notin \mathcal{H}_1(v)\}$
9:   Mark all hexagons as uncovered
10: while $\mathcal{H}_2(v) \neq \emptyset$ do
11:   $u \leftarrow$ a node in $\mathcal{H}_1(v)$ covering most uncovered hexagons (break ties based on smaller ID)
12:   for each uncovered hexagon covered by $u$ do
13:     $w \leftarrow$ a node with smallest ID among nodes in this uncovered hexagon and $N_1(u)$
14:     $\mathcal{H}_2(v) \leftarrow \mathcal{H}_2(v) \setminus \{\mathcal{H}_1(w) \cap \mathcal{H}_2(v)\}$
15:     $E_b \leftarrow E_b \cup \{(u, w)\}$
16:     Mark this uncovered hexagon as covered
17:   end for
18: end while
19: end for
20: $V_b \leftarrow$ nodes in $E_b$
ple only considers the broadcast structure of the hexagon with color 5. Recall that nodes in the same hexagon produce the same broadcast structure $T_b$. Hence, Algorithm 6.2 is illustrated from the perspective of node $v_1$. It starts by constructing the set $H_1(v_1)$ and $H_2(v_1)$ based on its two-hop neighbors information. Hence, $H_1(v_1) = \{v_1, v_2, v_3\}$ and $H_2(v_1) = \{v_4, v_5, v_6, v_7, v_8, v_9, v_{10}\}$. Node $v_1$ is first selected as the provider from $H_1(v_1)$ as it covers the most uncovered hexagons, i.e., hexagons with node $v_4, v_5, v_6$ and $v_{10}$. Next, nodes $v_4, v_5, v_6$ and $v_{10}$ are selected as the corresponding receptors of $v_1$, and are removed from $H_2(v_1)$, i.e., $H_2(v_1) = \{v_7, v_8, v_9\}$. Algorithm 6.2 then marks the hexagons covered by $v_1$ as covered, and adds into the set $E_b$ the links $(v_1, v_4)(v_1, v_5)(v_1, v_6)$ and $(v_1, v_{10})$. The other nodes in $H_1(v_1)$, i.e., $v_2$ and $v_3$, are handled in a similar manner and the final result is shown in Figure 6.3. For the hexagon with color 5, the set of providers is \{v_1, v_2, v_3\}.

The aforementioned $T_b$ construction process yields the following property.

**Lemma 6.2.** For each hexagon $H$, there are at most 18 providers with corresponding receptors located in $H$.

**Proof.** Recall that a provider $u$ and its corresponding receptor $v$ are located in different hexagons and only one link $(u, v)$ exists in $E_b$ between these two hexagons. This means it only needs to prove the number of hexagons covered by receptors in a given hexagon $H$ is upper-bounded by 18. As shown in Figure 6.2, for a given hexagon
$H$ with radius of $r/2$, it has at most 18 hexagons around it with a minimum distance less than $r$; that is, nodes in $H$ can cover at most 18 hexagons. Hence, this lemma holds.

6.2.2 Broadcast Scheduling

This section describes the protocol used to broadcast a message from the source node $s$ to all other nodes in $G_T$; see Algorithm 6.3.

HBA schedules the transmissions of nodes in $G_T$ in two phases. In Phase 1, the algorithm only considers nodes in $V_b$. Specifically, for each hexagon, denoted by $H$, HBA schedules the transmission from a provider $u$ to its corresponding receptor $v$ in $H$, where $(u, v) \in E_b$. In Phase 2, HBA allows a receptor $v$ in $H$ to transmit a broadcast message received in Phase 1 to all other nodes in $H$. Furthermore, HBA schedules all transmissions based on hexagons’ color, where those with a different color are not permitted to transmit or receive simultaneously.

Time is divided into different frames. A hexagon with the color value of $i$ is assigned to the $i$-th frame, denoted by $F_i$, where $1 \leq i \leq 3k^2$; recall that $3k^2$ is the number of colors used by Algorithm 6.1. As shown in Figure 6.4(a), each frame $F_i$ consists of three sub-frames, i.e., $F_i^1, F_i^2$ and $F_i^3$, comprising of $3T$ time slots. For each hexagon $H$, the first $T$ time slots, i.e., sub-frame $F_i^1$, is used to determine which provider $u$ from other hexagons is allowed to send a broadcast message $m$ to its corresponding receptor $v$ in $H$. Sub-frame $F_i^2$ is used to transmit message $m$ from provider $u$ to receptor $v$ in $H$. The last sub-frame, $F_i^3$, is used by receptor $v$ in $H$ to send the broadcast message to all other nodes in $H$. Specifically, Phase 1 is conducted in $F_i^1$ and $F_i^2$, and Phase 2 is carried out in $F_i^3$.

Let $c$ be the color of the hexagon containing source node $s$. Therefore, node $s$ initiates the broadcast by transmitting a message $m$ to all nodes in its hexagon in frame $F_c$. After that, HBA, see Algorithm 6.3, starts from frame $F_i$, where $i$ is initially set to $(c + 1) \mod 3k^2$; that is, it starts from the next frame of $F_c$ (line 4 in Algorithm 6.3). Then, HBA repeats the following iterations until all nodes in the network receive the broadcast message.
In Phase 1, for each hexagon $\mathcal{H}$ assigned to frame $\mathcal{F}_i$, i.e., they have color $i$, let $S_r(t)$ denote the set of receptors in $\mathcal{H}$ with active time slot $t$, and $S_p(t)$ denotes the set of providers that have received broadcast message $m$ before but have yet to send $m$ to their corresponding receptors in $S_r(t)$ (line 10 and 11 in Algorithm 6.3), where $1 \leq t \leq T$. Recall that these providers will have to wake-up at time $t$ to communicate with the receptors in $S_r(t)$. Let $\mathcal{F}^1_i(t)$ be the time slot $t$ of sub-frame $\mathcal{F}^1_i$, where $1 \leq t \leq T$. Denote by $w$ a receptor in $\mathcal{H}$ that received a REQUEST message from its corresponding provider before $\mathcal{F}^1_i(t)$, and $w$ is initially set to null. For any receptor $v \in S_r(t)$, $v$ first listens to the channel for an ACK message from a receptor $w$ when it wakes up at time slot $\mathcal{F}^1_i(t)$ (line 12 in Algorithm 6.3). This ACK is sent in the $\mathcal{AF}$ slot; see Figure 6.4(b). Then, for any provider $u \in S_p(t)$, it will send a REQUEST message to its receptor $v \in S_r(t)$ asking it to receive a broadcast message in sub-frame $\mathcal{F}^2_i$ (line 13 in Algorithm 6.3).

When receptor $v$ receives the REQUEST message from provider $u$, if $v$ has not received any ACK message from other receptors in $\mathcal{H}$, node $v$ replies with an ACK message to $u$. Otherwise, it does not respond to REQUEST messages (line 14 in Algorithm 6.3). As shown in line 16 of Algorithm 6.3, the selected receptor $v$ is assigned to $w$, and it is responsible for sending an ACK message in subsequent $\mathcal{AF}$ slots in $\mathcal{F}^1_i(t)$ (line 12 in Algorithm 6.3). This ensures all nodes waking up in subsequent slots are aware that a $w$ node is available, and thus stop responding to
The next issue is how providers in $S_p(t)$ transmit their REQUEST message in an interference free manner. Each time slot in $F^t_1$ is further divided into two parts, $AF$ and $BF$; see Figure 6.4(b). As mentioned earlier, $AF$ is used by receptors in $S_r(t)$ to listen to the channel for an ACK message, and for node $w$, if present, to transmit an ACK. The second part, namely $BF$, is divided into 18 sub-time slots, which is equal to the number of hexagons around $H$ that have a minimum distance less than $r$; cf. Lemma 6.2. These 18 sub-time slots are allocated to these neighboring hexagons according to their ID or color, which is known by every node. Hence, a provider $u \in S_p(t)$ is only able to send a REQUEST message to its corresponding receptor $v \in S_r(t)$ in the sub-time slot corresponding to its hexagon. A receptor $v$ is then required to reply with an ACK message in the same sub-time slot. As shown in Figure 6.4, each sub-time slot is sufficient to receive a REQUEST message and transmit an ACK message for receptor $v$, the duration of which is very short.

During time slot $t$ of sub-frame $F^2_t$, denoted by $F^2_t(t)$, where $1 \leq t \leq T$, the broadcast message $m$ is transmitted from the provider $u$ to receptor $v$ with active time slot $t$ (line 15 in Algorithm 6.3). Note, only one provider $u$ is selected in $F^1_t$ to relay the broadcast message in sub-frame $F^2_t$ to hexagon $H$.

In Phase 2, after receiving the broadcast message $m$, receptor $v$ will broadcast message $m$ to all other nodes in the same hexagon as $v$, i.e., $H$, in sub-frame $F^3_t$. The broadcast is carried out when these nodes wake up (line 19 in Algorithm 6.3). Finally, HBA updates $i$ to $(i + 1) \mod 3k^2$, and repeats the above steps until all nodes receive the broadcast message $m$.

The operation of Algorithm 6.3 is illustrated using Figure 6.3. Consider the hexagon with color 8. Assume receptor $v_6$, $v_{11}$ and $v_{12}$ have the same active time slot of $t$, and their corresponding providers, $v_1$, $v_{13}$ and $v_{14}$, have received a broadcast message $m$, and have yet to send to $v_6$, $v_{11}$ and $v_{12}$. Hence, as per Algorithm 6.3, $S_r(t) = \{v_6, v_{11}, v_{12}\}$ and $S_p(t) = \{v_1, v_{13}, v_{14}\}$. Nodes in $S_r(t)$, which are in hexagon 8, execute Algorithm 6.3 in frame $F_8$. In the $AF$ sub-time slot of $F^1_8(t)$, receptors in $S_r(t)$ listen to the channel for an ACK message. Suppose that no ACK message is sent at $AF$ by a node $w$. Also, in this case assume that the transmitting order of
Algorithm 6.3 Broadcast Scheduling

1: \textbf{input:} \( s, T_b = (V_b, E_b) \) and message \( m \)
2: \textbf{output:} Broadcast latency \( \text{Lat} \)
3: Schedule node \( s \) to transmit message \( m \) in frame \( F_c \)
4: \( \text{Lat} \leftarrow 0, V \leftarrow \mathcal{H}_1(s) \) and \( i \leftarrow (c + 1) \mod 3k^2 \)
5: \textbf{while} \( V \neq V_T \) \textbf{do}
6: \hspace{1em} // Phase 1-schedule nodes in \( V_b \)
7: \hspace{2em} \textbf{for} each hexagon \( \mathcal{H} \) assigned to frame \( F_i \) \textbf{do}
8: \hspace{3em} \( w \leftarrow \text{NIL} \)
9: \hspace{3em} \textbf{for} \( t \leftarrow 1 \) to \( T \) \textbf{do}
10: \hspace{4em} \( S_r(t) \leftarrow \{ v \mid v \text{ is a receptor in } \mathcal{H} \text{ with active time slot of } t \} \)
11: \hspace{4em} \( S_p(t) \leftarrow \{ u \mid u \text{ is a provider with } m \text{ and yet to send } m \text{ to its receptor } v \in S_r(t) \} \)
12: \hspace{4em} Node \( w \) sends an ACK at \( \mathcal{A}F \) of \( F_i^1(t) \)
13: \hspace{4em} \( S_p(t) \) sends a REQUEST at time allocated by \( \mathcal{H} \) among \( \mathcal{B}F \) of \( F_i^1(t) \)
14: \hspace{4em} \( v \in S_r(t) \) sends an ACK after receiving a REQUEST from its provider \( u \in S_p(t) \), if \( v \) has not heard an ACK from other receptors in \( \mathcal{H} \)
15: \hspace{4em} Node \( u \) sends message \( m \) to \( v \in S_r(t) \) at \( F_i^2(t) \)
16: \hspace{4em} \( w \leftarrow v \)
17: \hspace{3em} \textbf{end for}
18: \hspace{1em} // Phase 2-schedule nodes in \( V_T \setminus V_b \)
19: \hspace{2em} Node \( w \) broadcasts message \( m \) in \( F_i^3 \)
20: \hspace{2em} \( V \leftarrow V \cup \{ v \mid v \text{ is in } \mathcal{H} \} \)
21: \hspace{1em} \textbf{end for}
22: \hspace{1em} \( \text{Lat} \leftarrow \text{Lat} + 3T \) and \( i \leftarrow (i + 1) \mod 3k^2 \)
23: \textbf{end while}
providers is $v_1$, $v_{13}$ and $v_{14}$. As mentioned, the sub-time slots in $\beta \mathcal{F}$ can be assigned as per hexagon ID or color. In this example, provider $v_1$ first sends a REQUEST to $v_6$. On receiving this REQUEST, $v_6$ replies with an ACK immediately. After receiving this ACK from $v_6$, provider $v_1$ knows receptor $v_6$ is ready to receive the broadcast message $m$, and will transmit $m$ to $v_6$ in sub-frame $\mathcal{F}_8^2$. Other receptors, namely $v_{11}$ and $v_{12}$, will also receive the ACK from $v_6$, meaning they will not respond to any REQUEST from their respective provider; i.e., $v_{13}$ and $v_{14}$. At $\mathcal{F}_8^2(t)$, provider $v_1$ sends $m$ to $v_6$. In sub-frame $\mathcal{F}_8^3$, receptor $v_6$ broadcasts $m$ to nodes $v_{11}$ and $v_{12}$.

### 6.2.3 Distance-based Backoff

Recall that the $\beta \mathcal{F}$ portion of $\mathcal{F}_i^1(t)$ is divided into 18 sub-time slots. A possible optimization to shorten $\beta \mathcal{F}$ is by employing a distance-based backoff method. When a provider wants to send a REQUEST message, it first backs off for a period of time. This backoff duration depends on the distance between the provider and the hexagon containing its corresponding receptor. The smaller the distance, the shorter the backoff duration. Specifically, assume that a network operator decides to reduce the $\beta \mathcal{F}$ duration to $T_{\text{backoff}}$. This so called backoff time bound can be divided into $W \leq 18$ sub-time slots. Note, each sub-time slot is sufficient for transmitting an ACK and receiving a REQUEST message. Let $d$ be the distance between a provider $u$ and the centre of hexagon $\mathcal{H}$ containing node $u$’s receptor. It gets $d \geq \sqrt{3}r/4$ because $u$ is not included in $\mathcal{H}$, and the distance between $\mathcal{H}$’s edge and $\mathcal{H}$’s centre is $\sqrt{3}r/4$. Denote by $q$ the ratio between $\sqrt{3}r/4$ and $d$, i.e., $q = \frac{\sqrt{3}r}{4d}$, where $q \leq 1$. For provider $u$, it computes its backoff duration $t_{\text{backoff}}$ using the following equation,

$$
t_{\text{backoff}} = ([W(1-q)]) \frac{T_{\text{backoff}}}{W} + X \tag{6.3}
$$

where $X$ is random period of time generated from the range $[-\frac{T_{\text{backoff}}}{W}, \frac{T_{\text{backoff}}}{W}]$ for $1 \leq [W(1-q)] \leq W - 1$, and from range $[0, \frac{T_{\text{backoff}}}{W}]$ for $[W(1-q)] = 0$. The random value $X$ reduces the chance of interference when two or more providers have the same $q$. 
6.3 Analysis

The following set of theorems asserts the correctness, and approximation ratio of HBA in terms of the broadcast latency and transmission times.

**Theorem 6.2.** HBA yields a correct and interference-free broadcast schedule.

*Proof.* According to Theorem 6.1, transmissions are interference-free as long as the mutual distance between transmitters or receivers is larger than $\rho r$. Hence, as long as the simultaneous transmissions scheduled by HBA are separated by $\rho r$, the theorem is true.

Recall that in $G_T$, by design, the mutual distance between hexagons sharing the same color is larger than $\rho r$. For each frame $F_i$, only nodes in hexagons with the same color of $i$ are scheduled by HBA. Considering sub-frame $F_i^2$ and $F_i^3$, only providers and their corresponding receptors are allowed to send a broadcast message to nodes in hexagons with the same color of $i$. These receptors lie in hexagons with a color value of $i$, and hence, their mutual distance is larger than $\rho r$. Thus, these simultaneous transmissions during sub-frame $F_i^2$ and $F_i^3$ are interference-free by Theorem 6.1.

Next is to prove the transmissions in sub-frame $F_i^1$ are also interference-free. Only providers and their corresponding receptors are allowed to send a REQUEST and ACK during $F_i^1$. For hexagon $H$ with color value of $i$, the transmissions of providers and their receptors in $H$ are assigned with non-overlapping sub-time slots in $F_i^1$, and hence, transmissions in the same hexagon $H$ are interference-free. For different hexagons with color $i$, the mutual distance of receptors lying in them is lower-bounded by $\rho r$. According to Theorem 6.1, simultaneous transmissions in different hexagons during sub-frame $F_i^1$ are also interference-free.

**Theorem 6.3.** HBA produces a constant approximation for the MLBS problem with a ratio of $9 \left(\frac{2(\rho + 2)}{3}\right)^2 T$ in terms of broadcast latency.

*Proof.* The theoretical lower bound of the MLBS problem is $R$, i.e., the radius of the network with respect to the source node $s$. To compare the broadcast latency of HBA algorithm with the theoretical lower bound $R$, consider the BFS tree of the transmission graph $G_T$ rooted at $s$. This tree divides the network into layers $L_1, L_2, \ldots, L_R$. Let $t_i$ denote the maximum reception time of nodes in $L_i$, where $1 \leq i \leq R$. Then, to prove the correctness of this theorem, it is first to prove the following property, for each layer $L_i$, $t_i \leq t_{i-1} + 9 \left(\frac{2(\rho + 2)}{3}\right)^2 T$. The proof is by induction.
Layer $L_1$ only contains source node $s$, and thus $t_1 = 0$. Nodes in $L_2$ are the one-hop neighbors of $s$. Thus, for layer $L_2$, this property also holds true because node $s$ takes a frame, i.e., $3T$, to broadcast the message $m$ to $L_2$, i.e., $t_2 = t_1 + 3T$. Next is to prove this property is also true for layer $i$, where $3 \leq i \leq R$. Recall that nodes in $L_i$ are the one-hop neighbors of $L_{i-1}$. After $t_{i-1}$, receptors in $L_i$ will take at most $3 \left\lceil \frac{2(\rho + 2)}{3} \right\rceil^2$ frames to get the message $m$ from providers in $L_{i-1}$ and broadcast $m$ to other nodes in $L_i$, where $3 \left\lceil \frac{2(\rho + 2)}{3} \right\rceil^2$ is the maximum color number used by Algorithm 6.1. Note, a frame contains $3T$ time slots. Thus, for each layer $L_i$, $t_i \leq t_{i-1} + 9 \left\lceil \frac{2(\rho + 2)}{3} \right\rceil^2 T$. After $(R - 1)3 \left\lceil \frac{2(\rho + 2)}{3} \right\rceil^2$ frames, nodes in $L_R$ will receive the broadcast message $m$. Hence, the broadcast latency of HBA is upper-bounded by $(R - 1)9 \left\lceil \frac{2(\rho + 2)}{3} \right\rceil^2 T < 9 \left\lceil \frac{2(\rho + 2)}{3} \right\rceil^2 TR$.

**Theorem 6.4.** The number of REQUEST, ACK and broadcast messages in HBA is upper-bounded by $18N_H$, $TN_H$ and $(T + 1)N_H$ respectively, where $N_H$ is the number of hexagons required to cover the entire network.

**Proof.** Firstly, we need to show that the number of REQUEST messages is upper-bounded by $18N_H$. According to Lemma 6.2, for each hexagon $H$, it has at most 18 providers with corresponding receptors that are in $H$. Recall that a REQUEST message is only sent once from a provider to its corresponding receptor, which means, for each hexagon $H$, its receptors receive at most 18 REQUEST messages. Hence, given $N_H$ hexagons, the number of REQUEST messages is upper-bounded by $18N_H$.

Next is to show that the number of ACK messages is upper-bounded by $TN_H$. For each hexagon $H$, the ACK message is first sent by a receptor $v$ in $H$ in response to a REQUEST message from $v$’s corresponding provider. An ACK message is sent once in each time slot until sub-frame $F^1_i$ ends. Since $F^1_i$ consists of $T$ time slot, the maximum transmission time of ACK is $T$ for each hexagon. To sum up, the maximum number of ACK sent during the broadcast is $TN_H$.

Lastly, it remains to be proven that the maximum number of broadcast messages transmitted by HBA is $(T + 1)N_H$. As illustrated in Section 6.2.2, during sub-frame $F^2_i$ for a hexagon $H$, only one provider is allowed to transmit a broadcast message to its corresponding receptor $v$ in $H$. During each time slot of $F^2_i$, receptor $v$ will transmit a broadcast message at most $T$ times to its neighbors in $H$. The maximum number of broadcast messages transmitted in a hexagon is thus $T + 1$, meaning the total number of broadcast messages is upper-bounded by $(T + 1)N_H$.
6.4 Evaluation

This section outlines the research methodology used to evaluate the performance of HBA. In each experiment, each algorithm is measured against two metrics: broadcast latency and number of transmissions. In each experiment, the following parameters are set as follows: $\alpha = 4$, $\beta = 1$ and $r_{\text{max}} = 100$ meter. Wireless nodes are placed in a square area of $700 \times 700$ m$^2$ randomly. The following variables are studied: the number of nodes, transmission range $r$ and scheduling period $T$. For each experiment, one variable is varied whilst the other two remain unchanged. Each experiment is conducted on 50 randomly generated topologies. Moreover, for each topology, ten runs are conducted, where a source node is selected uniformly and randomly. Hence, each result is an average of $50 \times 10$ simulation runs.

6.4.1 Performance of HBA

![Figure 6.5: Broadcast latency under different number of nodes](image)

In Figure 6.5 and Figure 6.6, we delineate the broadcast latency and number of transmissions for different number of nodes, respectively. The value of $r$ is fixed at 50 meter, and $T$ is set to 10. As shown in Figure 6.5, we see that the broadcast latency of HBA decreases as the number of nodes increases. The reason is as follows. For a fixed area, the network becomes denser when the number of nodes becomes larger.
As a result, there are more links in the network, and thus the path from the source to the furthest node becomes shorter. However, from Figure 6.6, we observe that the number of transmissions increases with higher number of nodes. The reason is that more hexagons will be filled with nodes when the number of nodes becomes larger, and hence, more transmissions are required to propagate the broadcast message to these hexagons.

Figure 6.7 and 6.8 show the performance of HBA under different $r/r_{max}$, where $r$ is the transmission range in $G_T$, and $r_{max}$ is the maximum transmission range with a fixed value of 100 meter. In this experiment, the number of nodes is fixed to 400, $T$ is set to 10, and $r$ ranges from 50 to 90 meter. As shown in Figure 6.7, the broadcast latency decreases when $r$ ranges from 50 to 80 meter. This is because a larger transmission range $r$ leads to more links, and higher connectivity. As a result, HBA is able to find shorter broadcast paths. On the other hand, according to Theorem 6.1, a larger $r$ also prevents more nodes from transmitting simultaneously, which will result in longer broadcast paths. Hence, when the transmission range $r$ exceeds 80 meter in Figure 6.7, the broadcast latency starts to increase. We see that in Figure 6.8 the number of transmissions decreases with increasing $r$. This is because the number of hexagons used to cover the network becomes smaller when $r$ becomes larger. Hence, there are fewer transmissions when $r$ is large.
Figure 6.7: Broadcast latency under different range $r$

Figure 6.8: Number of transmissions under different range $r$
**Figure 6.9:** Broadcast latency under different duty cycle

**Figure 6.10:** Number of transmissions under different duty cycle
Figure 6.9 and 6.10 depict the performance of HBA for different scheduling periods $T$. The value of $T$ ranges from 5 to 25 ($r = 50$ meter and the number of nodes is set to 400). Note that the broadcast latency and number of transmissions increase with higher $T$ values. A larger $T$ will result in larger frames, and thereby, leads to higher broadcast latency. Consequently, for each hexagon, a receptor needs to transmit more times to its neighbors with different active time slots.

### 6.4.2 Performance of HBA versus Tree-based Algorithm

In this section, HBA is compared against the Tree-based algorithm [40] for always-on wireless networks. Recall that the Tree-based algorithm [40] is the first centralized method designed for always-on networks under the SINR-based interference model. In this respect, HBA is the first distributed algorithm designed for always-on and duty-cycled WSNs under the SINR-based interference model. In order to compare HBA faithfully against the Tree-based algorithm [40], the scheduling period used by HBA is set to 1, i.e., $T = 1$, meaning HBA also works in the always-on mode.

As shown in Figure 6.11(a), the broadcast latency of the Tree-based algorithm is around two times larger than that of HBA. This is mainly because the Tree-based algorithm is conducted layer by layer based on the BFS tree and nodes in lower layers are prevented from transmitting until all nodes in the current layer have finished their transmissions even though these transmissions do not cause any interference; instead, transmissions in HBA is handled in a greedy manner, which allow nodes to transmit as long as they do not result in interference. In addition, we observe that the number of transmissions experienced by nodes using HBA is about 50% larger than that of the Tree-based algorithm. This is because the Tree-based algorithm selects a minimal CDS as transmitting nodes in a centralized manner, which reduces the number of transmissions efficiently. However, HBA is a distributed algorithm with the local knowledge of two hops neighbors, and is not able to reduce redundant transmissions efficiently, compared with a centralised method such as [40].
Figure 6.11: Performance of HBA versus Tree-based Algorithm
6.5 Conclusion

This chapter studied the MLBSDC problem under the physical interference model. To achieve interference-free broadcast with the minimum broadcast latency, a novel algorithm, called HBA, is designed for nodes that employ a random duty-cycle schedule. Indeed, HBA is the first algorithm for the MLBSDC problem under the physical interference model. This chapter proves that HBA gives a correct and interference-free schedule, and produces a low broadcast latency in low overheads. Extensive simulation results show that HBA has better performance in terms of the broadcast latency than the Tree-based algorithm [40].
Conclusions

This thesis has developed algorithms for the MLBSDC problem under three different interference models, i.e., the RTS/CTS, protocol and physical interference model. As shown in Chapter 1, the MLBSDC problem is of great importance to applications such as disaster relief, the military, rescue operation or object detection, all of which impose stringent latency requirements. Moreover, in the context of duty-cycled WSNs, nodes have asynchronous wake-up times, and hence, not all neighbors of a transmitting node will receive a broadcast message at the same time, meaning multiple transmissions may be necessary. To this end, this thesis contributes to state-of-the-art by addressing the MLBSDC problem under different interference models in the following manner.

Chapter 3 studies the MLBSDC problem under the RTS/CTS interference model, and presents two tree-based algorithms: BS-1 and BS-2. This chapter proves that BS-1 and BS-2 provide correct and collision-free broadcast schedules, and produces low broadcast latencies and low overheads. Furthermore, BS-2 achieves the best constant approximation ratio in terms of broadcast latency, as compared to OTAB, the state of the art algorithm reported in [46]. Instead of utilizing the layer-by-layer approach, which is adopted by most existing tree-based algorithms, see Table 2.4, BS-1 and BS-2 speed up the broadcast by allowing transmissions across different layers simultaneously. Finally, extensive simulation results show that BS-1 and BS-2 have much better performance in terms of broadcast latency and number of transmissions.
Chapter 4 outlines a greedy heuristic algorithm and its distributed implementation, CEN and DIS respectively, for the MLBSDC problem under the RTS/CTS interference model. CEN and DIS are designed for nodes that employ pseudo-random MAC models such as ArDeZ [15], which ensures all neighbors of a broadcasting node are awake to receive a broadcast message. They consider the set of all nodes that have received the broadcast message as potential transmitters at any point in time. Consequently, from the viewpoint of the broadcast tree, CEN and DIS are able to schedule simultaneous transmissions from multiple layers, thereby, speeding up the propagation of a broadcast message. In particular, they apply two schemes, MTS and MITS, to select a maximal subset of collision-free transmitters. Specifically, the MTS algorithm selects nodes that have good coverage, whilst MITS ensures transmitting nodes are scheduled at non-conflicting times. The simulation results show that both CEN and DIS have near optimal performance in terms of the broadcast latency and have lower redundancy of transmissions than improved flooding and OTAB under different network scenarios.

Chapter 5 studies the MLBS problem under the protocol interference model and proposes two constant approximation algorithms: IABBS and IAEBS. This thesis proves that IABBS and IAEBS provide correct and interference-free broadcast schedules, and produces low broadcast latencies. Unlike past works, which adopt a stronger geometrical constraint to schedule interference-free transmissions, IABBS and IAEBS allow two simultaneous transmissions to proceed only if neither senders are located within the interference range of their corresponding receiver. This in turn helps to produce more interference-free transmissions, and reduce the broadcast latency. This chapter confirms that the latencies achieved by IABBS and IAEBS are much lower than existing schemes. In particular, compared to CABS [55], the best constant approximation broadcast algorithm to date, the broadcast latency achieved by IAEBS is only $\frac{5}{8}$ that of CABS. Furthermore, this chapter also shows that both IABBS and IAEBS are also applicable for duty-cycled WSNs, and proves they also give a solution with a constant approximation ratio for the broadcast latency and number of transmissions.
Chapter 6 presented a distributed greedy heuristic algorithm, called HBA, for the MLBSDC problem under the physical interference model. It then proves HBA gives a correct and interference-free solution with a constant approximation ratio for the broadcast latency. Moreover, HBA is the first work to study the MLBSDC problem under the physical interference model. HBA shows that cumulative interference from other simultaneous senders can be neglected if the mutual distance between a sender and other simultaneous senders exceeds some value, i.e., $\rho r$. As a result, HBA only allows nodes with the mutual distance larger than $\rho r$ to transmit or receive at the same time. Extensive experimental results show that on average, HBA has much better performance, i.e., $\frac{1}{2}$, in terms of broadcast latency than the tree-based algorithm reported in [40].

A key future research direction is to implement the algorithms proposed in this thesis in a more realistic, probabilistic interference model, where a message can be received successfully with varying probabilities as per SINR levels. Under deterministic interference models, e.g., the RTS/CTS, protocol and physical interference model, message reception is considered successful when there are no interfering nodes located in a receiver’s interference range or the SINR level of a message is higher than a threshold, i.e., $\beta$. However, such deterministic interference models do not reflect the probabilistic behaviour of wireless communications in the real world, where the success probability of message reception depends on the SINR level. That is, the higher the SINR, the higher the reception probability. The main challenge when adopting such probabilistic interference models is that a message may be retransmitted multiple times before reaching its intended receiver due to the fact that the success possibility is generally less than ‘1’, which gives rises to high broadcast latency.

Another research direction is to implement the tree-based algorithms introduced in Chapter 3 and 5 in a distributed manner. The distributed implementation of a tree-based algorithm for the MLBS problem is still an open problem. To date, as shown in Table 2.4, there is no such distributed tree-based algorithm. The main challenges are to develop distributed algorithms for the following problems: (1) determining the MIS set based on a BFS or SPT tree, and (2) assigning nodes with interference-free transmission time using only local topology information. The use of the methods
reported in this thesis for the all-to-all broadcast problem is another possible future work, where each node needs to propagate messages to all other nodes in the network.
Bibliography


Appendix

A.1 Correctness Analysis of BS-1 in Chapter 3

Theorem A.1. BS-1 provides a correct and collision-free broadcast schedule.

Proof. The correctness for this theorem is proven by contradiction. It is assumed that node \( v \) in layer \( i \) cannot be scheduled to receive the message collision-free because there are two or more parallel transmissions to node \( v \) at the same time. Assume that node \( v \)'s parent \( P(v) \) and one of \( v \)'s neighbor \( u \) are scheduled to transmit at \( t \). Furthermore, two different cases are taken into account. In the first case, node \( P(v) \) is scheduled before node \( u \). According to constraint 4) of Algorithm 3.2, if node \( P(v) \) selects time \( t \) as \( P(v) \)'s transmission time, node \( u \) will not choose \( t \) again, because the reception time of its neighbor \( v \) has been already set to \( t \). Secondly, assume that node \( u \) is scheduled before node \( P(v) \). According to constraint 3) of Algorithm 3.2, after node \( u \) selects time \( t \) as its transmission time, node \( P(v) \) will not choose \( t \) as its transmission time since node \( P(v) \)'s children \( v \) will hear node \( u \)'s transmission at \( t \). This is contradictory to the assumption. So node \( v \) will receive the message collision-free.

It now proves BS-1 produces a schedule with 100% reachability by contradiction. Assume there exists a node \( v \) that has not received a broadcast message, but all of node \( v \)'s neighbors are covered by BS-1. Recall that node \( v \) must be assigned with a parent node from its one-hop neighbors, and \( v \) can receive the message collision-free from its parent node as proven above. Therefore, this case is contradictory to the assumption. Hence, this theorem holds true for all nodes in the network.
A.2 Approximation Ratio Analysis of BS-1 in Chapter 3

Lemma A.1. Consider a parent $u$ of nodes in $L_i$. Suppose that node $u$'s transmission to layer $L_i$ is delayed because doing so will cause a collision at one or more of its one hop neighbors, denoted by set $W$, with transmissions among its two-hop neighbors. Then, the following is true:

1. Node $w$ is not in $C(u, i)$, where $w \in W$.
2. For node $u$, there are at most $\Delta - 2$ nodes among its two-hop neighbors that interfere with $u$.

Proof. The correctness of the first property is proven by contradiction. Assume that node $w$ is in $C(u, i)$, and there is a node $z$ that interferes with $u$ at $w$. According to the construction of $T_b$, node $z$ must have one child node in layer $i$; that is, $z$ must have one child which has the same active time slot with nodes in $C(u, i)$, and is scheduled to transmit to layer $i$ before node $u$. Recall that the order in which nodes’ parent nodes are selected (line 7 in Algorithm 3.1) and the order in which the transmissions are scheduled (line 5 in Algorithm 3.3) are the same. That is, if node $z$ is scheduled to transmit to layer $i$ before node $u$, node $w$ must be node $z$’s child, i.e., $w \in C(z, i)$. This contradicts the assumption that node $w$ is in $C(u, i)$.

According to the first property of Lemma A.1, when node $u$ is scheduled to transmit to nodes in layer $i$, none of $u$’s children in $C(u, i)$ hear a message from other nodes; That is, $I_1(u) = \emptyset$ as per Algorithm 3.2. Hence, node $u$’s transmission time is only delayed by the reception time of its one-hop neighbors as per Algorithm 3.2, i.e., $I_2(u)$ . Recall that node $u$ has at most $\Delta$ one-hop neighbors, in which one is $u$’s parent node and at least one is $u$’s child. Henceforth, node $u$ is adjacent to at most $\Delta - 2$ interfering nodes that have been assigned with reception time when node $u$ is considered, i.e., $|I_2(u)| \leq \Delta - 2$. That is, node $u$’s transmission to layer $i$ is interfered by at most $\Delta - 2$ nodes.

Lemma A.2. Consider a parent $u$ of nodes in layer $L_i$. Let time $tr(u, i)$ be its scheduled transmission time. Then, $tr(u, i) \leq rec(u) + (\Delta - 1)T$.

Proof. According to Lemma A.1, $I_1(u) = \emptyset$ and $|I_2(U)| \leq \Delta - 2$. Recall that in Algorithm 3.2, $tr(u, i) = \min \{t | t \text{ mod } T = \tau_i, t > rec(u) \text{ and } t \notin I_1(u) \cup I_2(u)\}$, therefore $tr(u, i) \leq rec(u) + (\Delta - 2)T + T$, where $(\Delta - 2)T$ accounts for the delay incurred by transmissions from $\Delta - 2$ nodes to their children in layer $i$ in the worst case.
Lemma A.3. Denote by $r_i$ the maximum reception time of nodes from layer 0 to $i$. Then, for layer $i$, where $i > 0$, $r_i \leq (i - 1)(\Delta - 1)T$.

Proof. This lemma is proven by induction. For layer 1, it holds true because the reception time of nodes in layer 1 is 0. Next is to prove this lemma is also true for layer $i$, where $i > 1$. By Lemma A.2, the maximum transmission time of nodes from layer 0 to $i - 1$ is bounded by $r_{i-1} + (\Delta - 1)T$. Since nodes in layer $i$ get the message from the upper layers, $r_i$ is bounded by the maximum transmission time of nodes from layer 0 to $i - 1$, i.e., $r_i \leq r_{i-1} + (\Delta - 1)T \leq (i - 1)\Delta - 1)T$. Therefore, it is also correct for layer $i$.

Theorem A.2. BS-1 is an $(\Delta - 1)T$-approximate solution for the MLBSDC problem.

Proof. By Lemma A.3, the maximum reception time of nodes in $T_b$ is $r_1$, and its maximum value is $(l - 1)(\Delta - 1)T$. That is, BS-1 needs to take at most $(l - 1)(\Delta - 1)T + 1$ time slots to finish the broadcast, where 1 accounts for time slot 0. Recall that $l \leq H$, and therefore, in the worst case, the broadcast latency for BS-1 is $(l - 1)(\Delta - 1)T + 1 \leq (H - 1)(\Delta - 1)T + 1 < H(\Delta - 1)T$, which proves the theorem.

Theorem A.3. The total number of transmissions scheduled by BS-1 do not exceed $|V| - 1$.

Proof. Recall that the transmission of BS-1 is scheduled layer by layer in a top-down manner and each parent node is only allowed to transmit once to its children in a given layer. Therefore, for each layer $i$, it requires at most $|L_i|$ transmissions to inform all nodes in $L_i$. Hence, the number of transmissions performed by BS-1 is bounded by $\sum_{i=1}^{l} |L_i| = |V| - 1$.

A.3 Correctness Analysis of BS-2 in Chapter 3

Theorem A.4. BS-2 yields a correct and collision-free broadcast schedule.

Proof. Recall that BS-2 has two phases. This means it only needs to prove that the reception of all nodes in each phase is collision free. In each phase, BS-2 is conducted layer by layer with the same scheduling constraints as BS-1. Consequently, by the proof of Theorem A.1, BS-2 also yields a collision free schedule. Furthermore, as mentioned above, all dominators and their parent nodes in BS-2 form a CDS. Therefore, all nodes can receive the broadcast message from this CDS collision-free.
A.4 Approximation Ratio Analysis of BS-2 in Chapter 3

Lemma A.4. Consider a parent $u \in M$ of nodes in $L_i \cap X$. Suppose that in Phase 1, node $u$’s transmission to nodes in layer $i$ is delayed because doing so will cause a collision at one or more of its one-hop neighbors, denoted by set $W$, with transmissions from nodes in $M$ among its two-hop neighbors. Then the following is true.

1. Node $w$ is not in $C(u, i)$, where $w \in W$.

2. For node $u$, there are at most three nodes in $M$ among its two-hop neighbors that interfere with $u$.

Proof. Similar to Lemma A.1, the order in which the parent of nodes in $U$ is chosen (line 13 in Algorithm 3.4) and the order in which the transmissions are scheduled (line 6 in Algorithm 3.5) are the same. That is, if there is node $z \in M$ among $u$’s two-hop range that is scheduled to transmit to layer $i$ before node $u$, node $w$ should be in $C(z, i)$, not in $C(u, i)$. Hence, node $w$ must be not in $C(u, i)$.

According to the first property of Lemma A.4, when node $u$ is scheduled to transmit to nodes in layer $i$, none of $u$’s children in $C(u, i)$ hear a message from other nodes in $M$ among $u$’s two-hop range, i.e., $I_1(u) = \emptyset$ as per Algorithm 3.2. Hence, node $u$’s transmission time is only delayed by the reception time of its one-hop neighbors that receive the message from nodes in $M$ among $u$’s two-hop range. Recall that in $T_b$, the children (respectively, parent nodes) of nodes in $M$ are selected from dominators, i.e., $U$, and each node can be adjacent to at most five dominators, see [24]. Hence, there are at most $5 - 1$ dominators among $u$’s one-hop range that are assigned with reception time when $u$ is considered, where 1 accounts for at least one child of node $u$. Except $u$’s parent node that is also a dominator, node $u$ is adjacent to at most three interfering dominators that have been assigned with reception time when node $u$ is considered, i.e., $|I_2(u)| \leq 3$. In the worst case, three dominators have three different parent nodes in $M$, hence there are at most three nodes in $M$ among $u$’s two-hop neighbors that interfere with $u$. \hfill $\Box$

Lemma A.5. Consider a dominator $v \in U$ that is a parent of nodes in $L_i \cap X$. Suppose that in Phase 1, node $v$ had to defer its transmission to nodes in layer $i$ which otherwise would cause a collision at one or more of its one-hop neighbors, denoted by set $W$, with transmissions from dominators in $U$. Then the following is true.

1. Node $w$ is not in $C(v, i)$, where $w \in W$.

2. For node $v$, there are at most eight dominators in $U$ among its two-hop neighbors that interfere with $v$. 

Proof. Similar to Lemma A.1 and A.4, the order in which the parent nodes of nodes in $M$ are selected (line 19 in Algorithm 3.4) and the order in which the transmissions are scheduled (line 7 in Algorithm 3.5) are the same. That is, if there is dominator $z \in U$ among $v$'s two-hop range that is scheduled to transmit to layer $i$ before node $v$, node $w$ should be in $C(z, i)$, not in $C(v, i)$.

None of $v$'s children in $C(v, i)$ hear a message from other dominators in $U$ among $v$'s two-hop range, when node $v$ is scheduled to transmit to nodes in layer $i$, based on the first property. Hence, dominators among $u$'s two-hop range only collide with $u$ at its one-hop neighbors that are not its children. Recall that in $T_b$, the children (respectively, parent nodes) of dominators in $U$ are selected from nodes in $M$, and only non-leaf nodes in $M$ are scheduled in Phase 1. Moreover, the number of dominators among $v$'s two-hop range does not exceed 19, see [6]. Let $z \in L_{l \leq i}$ be a dominator among $v$'s two-hop range that is scheduled to transmit to its children $C(z, i)$ in Phase 1 before $v$. Since nodes in $C(z, i)$ belong to $M$, they must have at least one child that is also a dominator among $v$'s two-hop range in a lower layer than $i$, or else nodes in $C(z, i)$ are the leaf nodes in $T_b$, and $z$'s transmission will be scheduled in Phase 2, not in Phase 1. Dominators in lower layers than $i$ will not interfere with $v$, so for each $z$, there is at least one dominator among $v$’s two-hop range that does not interfere with $v$. Thus, at most half of dominators among $v$’s two-hop range are scheduled to transmit to layer $i$ before $v$, i.e., $\left\lfloor \frac{19}{2} \right\rfloor = 9$. Excluding one dominator that is the parent node of $P(v)$ and also among $v$’s two-hop range, node $v$’s transmission to layer $i$ is interfered by at most eight dominators among its two-hop range.

\[ \square \]

Lemma A.6. Denote by $r_i$ the maximum reception time of nodes from layer 0 to $i$ in Phase 1. Then, for layer $i$, where $0 < i < l$, $r_i \leq (i - 1)13T$.

Proof. This lemma is proven by induction. For layer 1, this lemma holds true because the reception time of nodes in layer 1 is 0. Then, it proves this lemma is also true for layer $i$, where $1 < i < l$. For each layer $i$ in Phase 1, the transmissions by parent nodes of nodes in $U_i$ are scheduled before parent nodes of nodes in $M_i \cap X$. Let node $u$ be a parent node of nodes in $U_i$. Suppose that schedule the transmissions of node $u$ after $r_{i-1}$, this means, only transmissions from parent nodes to nodes in $U_i$ will interfere with $u$’s transmission, because all nodes from layer 0 to $i-1$ receive the message by $r_{i-1}$ and the transmitters of nodes in $M_i$ are scheduled after node $u$. By Lemma A.4, node $u$’s children will not hear the message from other parent nodes of nodes in $U_i$, and there are at most three parent nodes of nodes in $U_i$ that are scheduled to transmit before $u$. Therefore, as per Algorithm 3.2, $|I_u| \leq 3$ and the maximum transmission time of node $u$ is $r_{i-1} + 4T$, i.e., $tr1(u, i) \leq r_{i-1} + |I(u)|T + T$.

Since all parent nodes of nodes in $U_i$ transmit by $r_{i-1} + 4T$, suppose a parent node $v \in U$ of nodes in $M_i \cap X$ is scheduled after time $r_{i-1} + 4T$. It means, only transmissions from parent nodes of nodes in $M_i \cap X$ interfere with $v$’s transmission, because all nodes in $U_i$ and layer 0 to $i-1$ receive the message by $r_{i-1} + 4T$. By Lemma
A.5, node $v$’s children will not hear the message from other parent nodes of nodes in $M_i \cap X$, and there are at most eight parent nodes of nodes in $M_i \cap X$ that are scheduled to transmit before $v$. Therefore, as per Algorithm 3.2, $|I_v| \leq 8$ and the maximum transmission time of node $v$ is $r_{i-1} + 13T$, i.e., $tr1(v,i) \leq r_{i-1} + 4T + |I(v)| + T$. Therefore, all nodes in layer $i$ will receive the message by $r_{i-1} + 13T$ in Phase 1, i.e., $r_i \leq r_{i-1} + 13T \leq (i-1)13T$. Hence, it is also correct for layer $i$.

Lemma A.7. In Phase 1, all nodes receive the broadcast message by $(l-2)13T + 4T$.

Proof. By Lemma A.6, the maximum reception time for nodes from layer 0 to $l-1$ is $(l-2)13T$. For layer $l$, only nodes in $U_l$ are scheduled to receive from their parent nodes in Phase 1, and thus if their parent nodes are scheduled after $(l-2)13T$, their transmissions are delayed by at most $4T$ time for the same reason as in Lemma A.6. Hence, in Phase 1, all nodes will receive the message by $(l-2)13T + 4T$.

Lemma A.8. Consider a node $u$ that is a member of $M_i \cap Y$, where $0 < i < l$. Then, $rec(u) \leq (l-2)13T + 24T$ in Phase 2.

Proof. After time $(l-2)13T + 4T$, node $u$’s transmission can only be corrupted by transmissions from dominators in Phase 2, because all nodes in $X$ receive the message by $(l-2)13T + 4T$ and only dominators are allowed to transmit in Phase 2. Therefore in Phase 2, node $u$ must receive the message collision-free from its parent node $v \in U$ if node $v$ avoids transmitting the message at the time when other dominators among node $v$’s two hops’ range transmit to node $u$. Similar to Lemma A.1, A.4 and A.5, the order in which the parent nodes are selected (line 19 in Algorithm 3.4) and the order in which the transmissions are scheduled (line 19 in Algorithm 3.5) are the same. Hence, $u$ will not hear a message from other nodes after time $(l-2)13T + 4T$ when $u$ is scheduled. Recall that the size of dominators in a radius two circle does not exceed 19, and thus the size of dominators that interfere node $u$’s transmission at $u$’s one-hop neighbors is not over 19, i.e., $|I(u)| \leq 19$ as per Algorithm 3.2. Hence $rec(u) \leq (l-2)13T + 4T + 19T + T$.

Theorem A.5. BS-2 provides a $13T$-approximate solution for the latency.

Proof. By Lemma A.8, all nodes receive the message by $(l-2)13T + 24T$. That is, BS-2 takes at most $(l-2)13T + 24T + 1$ time slots to finish the broadcast. Recall that $l \leq H$, and thus, in the worst case, the broadcast latency for BS-2 is $(l-2)13T + 24T + 1 \leq (H-2)13T + 24T + 1 \leq 13TH - 2T + 1 \leq 13TH$.

Theorem A.6. BS-2 is a $4(T+3)$-approximate solution in terms of number of transmissions.
Appendix

Proof. Recall that only dominators and non-leaf nodes in \( M \) transmit the message in Phase 1. The number of dominators transmitting the message in Phase 1 does not exceed \(|U| - 1\). The value of 1 accounts for one dominator that is located in the last layer of Phase 1 and does not retransmit the message. Node \( s \) does not have a parent node and each non-leaf node in \( M \) must have at least one dominator as their child, thus the number of non-leaf nodes in \( M \) transmitting the message in Phase 1 does not exceed \(|U| - 1\). Consequently, the total number of nodes transmitting in Phase 1 does not exceed \( 2(|U| - 1) \).

For Phase 2, node \( s \) does not need to transmit the message, so there are at most \((|U| - 1)\) dominators transmitting the message. Recall that the parent nodes of nodes in \( M_i \) are chosen from dominators in \( U_{j<i} \). That is, a given dominator \( u \in U_i \) that can be a parent of node \( v \) in \( M_{j\geq i} \). Owing to \( \text{Lat}(u, v) \leq T \), and dominator \( u \) only needs to transmit once to its children in the same layer, for any dominator \( u \), the number of transmissions to children nodes is at most \( T + 1 \) times. Hence, the total number of transmissions in Phase 2 do not exceed \((T + 1)(|U| - 1)\). Therefore, the total number of transmissions scheduled by BS-2 do not exceed \((T + 3)(|U| - 1)\), i.e., \( 2(|U| - 1) + (T + 1)(|U| - 1) \). Recall that the size of \( U \) does not exceed \( 4\text{opt} + 1 \) [79], where \( \text{opt} \) denotes the minimum number of transmissions, BS-2 is thus a \((T + 3)(4\text{opt} + 1 - 1) = 4(T + 3)\text{opt} \) solution.

\[ \square \]

A.5 Correctness of Theorem 6.1 in Chapter 6

Lemma A.9. Given a set \( L \) of links, if the mutual distance of senders in \( L \) are greater than \( \rho r \), set \( L \) is independent, where \( \rho = 1 + \left( \frac{8\beta}{1-(r/r_{\max})^\alpha} \left( \frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3 \right) \right)^{1/\alpha} \).

Proof. Let \((u, v)\) denote a link belonging to \( L \). With sender \( u \) as the centre, partition the senders in \( L \) into concentric rings \( \text{ring}^k \) with width \( \rho r \). Ring \( \text{ring}^0 \) contains all senders \( w \) of links in \( L \) satisfying \( k\rho r \leq d(w, u) < (k + 1)\rho r \). The first ring \( \text{ring}^0 \) only contains sender \( u \). Then consider all senders \( w \in \text{ring}^k \) for some integer \( k > 0 \).

First, consider the distance between any senders \( w \) in \( \text{ring}^k \) and \( u \). As per the construction of rings, it has \( d(w, u) \geq k\rho r \) for ring \( \text{ring}^k \). Note that, \( d(u, v) \leq r \) and \( \rho > 1 \). Applying the triangle inequality, the lower bound of \( d(w, v) \) for \( \text{ring}^k \) is,

\[
\begin{align*}
    d(w, v) &\geq d(w, u) - d(u, v) \\
    &\geq k\rho r - r \\
    &> (\rho - 1)kr
\end{align*}
\]

(A.1)
Next, observe that for any senders $w$ in $\text{ring}^k$, the disk centred at $w$ with a radius of $\frac{1}{2}\rho r$ is non-overlapping with other senders in $\text{ring}^k$, and such a disk is fully contained in an extended ring of $\text{ring}^k$, with an extra width of $\frac{1}{2}\rho r$ at each side of $\text{ring}^k$. Then, by referring to the ratio between the area of this extended ring and the disk with a radius of $\frac{1}{2}\rho r$, the number of senders contained in $\text{ring}^k$ is upper-bounded by $8(2k + 1)$ as per Eqn. A.2.

\[
\frac{\pi(k+3/2)^2(\rho r)^2 - \pi(k-1/2)^2(\rho r)^2}{\pi(1/2\rho r)^2} = 8(2k + 1)
\]

(A.2)

The total interference $I_k$ emanating from $\text{ring}^k$ is bounded by

\[
I_k = \sum_{w \in \text{ring}^k} P_d(w, v)^{-\alpha}
\leq 8(2k + 1)P((\rho - 1)kr)^{-\alpha}
\]

(A.3)

Summing up the total interferences $I$ over all rings yields

\[
I = \sum_{k=1}^{\infty} I_k = \sum_{k=1}^{\infty} 8(2k + 1)P((\rho - 1)kr)^{-\alpha}
\]

(A.4)

Recall that $d(u, v) \leq r$ and $N = P/\beta r_{\text{max}}^\alpha$, where $r_{\text{max}}$ is the maximum transmission range in the absence of interference. If $v$ successfully receives a message from $u$ if and only if the following condition holds:

\[
\begin{align*}
\text{SINR} &= \frac{P_d(u, v)^{-\alpha}}{I + N} \\
&\leq \frac{P r^{-\alpha}}{\sum_{k=1}^{\infty} 8(2k+1)P((\rho - 1)kr)^{-\alpha} + P/\beta r_{\text{max}}^\alpha} \\
&= \frac{\beta}{\sum_{k=1}^{\infty} 8\beta(2k+1)(\rho - 1)^{-\alpha}k^{-\alpha} + (r/r_{\text{max}})^\alpha} \\
&\leq \beta
\end{align*}
\]

(A.5)

According to Inequality A.5, such SINR is at least $\beta$ if and only if

\[
\sum_{k=1}^{\infty} 8\beta(2k+1)(\rho - 1)^{-\alpha}k^{-\alpha} + (r/r_{\text{max}})^\alpha \leq 1
\]

(A.6)
According to Riemann zeta function, \( \sum_{k=1}^{\infty} k^{-\alpha} \leq \frac{1}{\alpha-1} + 1 \), where \( \alpha > 2 \). Plugging this in, it has
\[
\sum_{k=1}^{\infty} 8(2k+1)k^{-\alpha} \leq 8\left(\frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3\right) \quad (A.7)
\]

According to Inequality A.6 and A.7, it has
\[
\sum_{k=1}^{\infty} 8\beta(2k+1)(\rho-1)^{-\alpha}k^{-\alpha} + \left(\frac{r}{r_{\text{max}}}\right)^{\alpha} \\
= 8\beta\left(\frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3\right)(\rho-1)^{-\alpha} + \left(\frac{r}{r_{\text{max}}}\right)^{\alpha} \leq 1 \quad (A.8)
\]

When \( \rho = 1 + \left(\frac{8\beta}{1-\left(\frac{r}{r_{\text{max}}}\right)^{\alpha}}\left(\frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3\right)\right)^{1/\alpha} \), inequality A.8 must hold; in other words, receiver \( v \) can receive the message successfully in \( L \). In conclusion, set \( L \) is independent.

\[\square\]

**Lemma A.10.** Given a set \( L \) of links, if the mutual distance of receivers in \( L \) is greater than \( \rho r \), set \( L \) is independent, where \( \rho = 1 + \left(\frac{8\beta}{1-\left(\frac{r}{r_{\text{max}}}\right)^{\alpha}}\left(\frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3\right)\right)^{1/\alpha} \).

**Proof.** Suppose that a link \((u,v)\) is in \( L \). With the centre as receiver \( v \), then divide the receivers in \( L \) into concentric rings \( \text{ring}^k \). Recall that the length of each link is upper-bounded by \( r \) in \( L \). For a sender \( w \) whose receivers lie in \( \text{ring}^k \), \( d(w,v) \) is lower-bounded by \( \rho kr - r \geq (\rho - 1)kr \). That is, for \( \text{ring}^k \), the distance between interfered sender \( w \) and receiver \( v \) is no smaller than \( (\rho - 1)kr \). Next, using the same argument as Lemma A.9, it gets that \( L \) is also independent. \[\square\]

**Theorem A.7.** Given a set \( L \) of links, in order for set \( L \) to be independent, it is sufficient to have:

1. The mutual distance of senders are all greater than \( \rho r \); OR
2. The mutual distance of receivers are all greater than \( \rho r \).

where \( \rho = 1 + \left(\frac{8\beta}{1-\left(\frac{r}{r_{\text{max}}}\right)^{\alpha}}\left(\frac{2}{\alpha-2} + \frac{1}{\alpha-1} + 3\right)\right)^{1/\alpha} \).

**Proof.** It is proved by Lemma A.9 and A.10. \[\square\]