TREE-BASED CHANNEL ASSIGNMENT SCHEMES FOR MULTI-CHANNEL WIRELESS SENSOR NETWORKS

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Çağlar Terzi

September, 2012

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. İbrahim Körpeoğlu (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Özgür Ulusoy

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. A. Enis Çetin

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

TREE-BASED CHANNEL ASSIGNMENT SCHEMES FOR MULTI-CHANNEL WIRELESS SENSOR NETWORKS

Çağlar Terzi
M.S. in Computer Engineering
Supervisor: Assoc. Prof. Dr. İbrahim Körpeoğlu
September, 2012

A lot of sensor node platforms used for establishing wireless sensor networks (WSNs) can support multiple radio channels for wireless communication. Therefore, rather than using a radio single channel and sharing it for the whole network, multiple channels can be utilized in a sensor network simultaneously to decrease the overall interference in the network, which may help increasing the aggregate throughout in the network and decrease packet collisions and delay. This requires, however, appropriate channel assignment schemes to be used for assigning channels to the nodes for multi-channel communication in the network. Since, data generated by sensor nodes are usually carried to one or more sinks in the network using routing trees, tree-based channel assignment schemes are a natural approach for assigning channels in a WSN. We present two fast tree-based channel assignment schemes (called BUCA and NCCA) for multi-channel WSNs. We also propose a new network interference metric that is used in our algorithms while making decisions. We evaluate our proposed schemes by extensive simulation experiments and compare them with another well-known tree-based protocol from the literature. The results show that our proposed algorithms can provide better performance, up to 40% performance increase in some cases, compared to the other method. We also discuss in which cases the performance improvement can be achieved.

Keywords: Wireless sensor networks, multi-channel, tree-based channel assignment.

ÖZET

ÇOK-KANALLI KABLOSUZ ALGILAYICI AĞLARI İÇİN AĞAÇ-TABANLI KANAL ATAMA YÖNTEMLERİ

Çağlar Terzi
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assoc. Prof. Dr. İbrahim Körpeoğlu
Eylül, 2012

Kablosuz algılayıcı ağları (KAA) kurmak için kullanılan birçok algılayıcı düğüm düzlemleri kablosuz iletişim için birden çok radyo kanalını destekleyebilmektedir. Bu yüzden, tek bir radyo kanalı kullanıp tüm ağ için onu paylaşmak yerine, toplam üretilen işi arttırmaya ve paket çarpışmasını ve gecikmeyi azaltmaya yardımcı olabilecek toplam girişimi azaltmak için, eş zamanlı olarak birden çok kanaldan algılayıcı ağlarında yararlanılabilir. Bu ancak, çok-kanallı iletişimde kullanılacak düğümlere kanal atamak için uygun kanal atama yöntemleri kullanılmasını gerektirir. Genellikle algılayıcı düğümlerinde oluşturulan veri, ağda bulunan bir ya da daha çok alıcıya yol atama ağaçlarıyla taşındığı için, KAA'da kanal atamak için ağaç-tabanlı kanal atama yöntemlerinin kullanılması doğal bir yaklaşımdır. Çok-kanallı KAA'lar için iki adet hızlı ağaç-tabanlı kanal atama yöntemi sunulmaktadır (BUCA ve NCCA). Ayrıca, algoritmalarımızda karar verirken kullanılan yeni bir ağ girişim ölçevi de önerilmektedir. Önerilen yöntemler ayrıntılı benzetim ve deneylerle değerlendirilmektedir ve literatürde iyi bilinen başka bir ağaç-tabanlı iletişim kuralı ile karşılaştırılmaktadır. Sonuçlar önerdiğimiz algoritmaların diğer yöntem ile karşılaştırıldığında daha iyi başarım gösterdiğini ve %40'a kadar başarım artışı olduğunu göstermektedir. Ayrıca, hangi durumlarda başarım artışına ulaşıldığı tartışılmaktadır.

Anahtar sözcükler: Kablosuz algılayıcı ağları, çok-kanallı, ağaç-tabanlı kanal atama.

Acknowledgement

First of all, I am grateful to my supervisor, Assoc. Prof. Dr. İbrahim Körpeoğlu, for his support and guidance throughout the thesis process. I could not write this thesis without his assistance and encouragement. I learned a lot from him, and it was a pleasure for me to work with him and to be his student, I want to thank him again for all the help given.

I would like to thank to Prof. Dr. Özgür Ulusoy for being a great teacher and Prof. Dr. A. Enis Çetin for being a great leader. They were very kind to accept being a jury member and spend their valuable time for evaluating my thesis.

I would like to express my gratitude to all of my teachers for their help and contributions to my success. I also want to thank to all of my friends who support me and make me happy.

I am very happy to meet my office friends, especially those in Wireless Lab Group, and thank them for their support and enjoyable time we had together.

I would like to thank to Esragül Katırcıoğlu for her encouragement and support in every part in my thesis process and my life. She has considerable amount of contributions in this thesis.

Last but not least, I am grateful to my family members, especially my mother, my father, my sister and my maternal aunt, for their limitless support at every moment of my life. They sacrifice so much for me and provide the best opportunities. The thesis would be a dream without them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This thesis is about channel assignment in multi-channel wireless sensor networks (WSNs), which have applications in a variety of sectors including industrial automation, environment, health care, and military [5, 6]. A WSN is composed of a large number of sensor nodes, which are capable of sensing physical or environmental conditions such as temperature, humidity, sound, light, pressure, etc.

Sensor nodes have typically six parts: energy source, processor, memory, storage, sensor boards and a Radio Frequency (RF) transceiver. Energy source is the power source of the sensor nodes, and it generally consists of batteries. Energy consumption is the main problem of the sensor networks because it is hard to replace the batteries for WSNs that consist of large number of sensor nodes, which are scattered in a large area. Processors of the sensor nodes are low power processors and memories of sensor nodes are limited with a few kilobytes, and flash memories are used for storage, which are in the scale of MBs in the newer sensors. Sensor boards are used for sensing the environment, and half-duplex Radio Frequency (RF) transceiver is used for wireless communication among sensor nodes and/or other devices. One example of a sensor node can be seen in Figure 1.1.

A WSN consists of a large number of sensor nodes and one or more sink nodes. Sink nodes are sometimes referred as gateway nodes or base stations (BSs), which are the nodes in the network, where all other sensor nodes try to send data to.

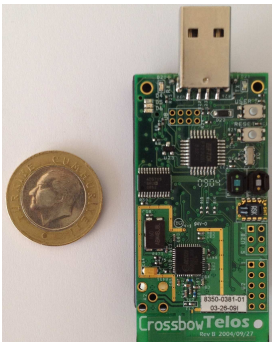

Figure 1.1: A TelosB [1] sensor node.

They are generally connected to a computer or Internet and controlled from there. Usually there is a single sink in a WSN, however, there can be more than one sink node in a WSN in order to reduce the energy consumption or increase the manageability of the network [7].

Sensor nodes communicate with each other by choosing a channel (frequency) in their defined frequency domain. ZigBee-based sensor nodes can operate on 868 MHz (1 channel), 915 MHz (10 channels) or 2.4 GHZ (16 channels) Industrial Scientific and Medical (ISM) bands. ZigBee is a specification developed for low-power radios, which is based on IEEE 802.15.4 standard for low-rate wireless personal area networks (PANs) [8, 9].

In single channel communication, all of the sensors in the WSN use the same channel, which causes high interference for that network and decreases the throughput. To decrease the interference and provide parallel communication, newer sensors like Chipcon CC2420 [10] and Crossbow TelosB [1] provide multi-channel communication by using ZigBee/802.15.4 standard. Most sensor nodes have half-duplex radios, which means that a sensor node is able to transmit or receive data, but not at the same time.

In multi-channel communication, nodes in a WSN can use different channels. Each node, however, can operate in one channel at a time if they do not have multiple radios. Two nodes (i.e., a node pair) that want to communicate have to use the same channel. The main aim of using multi-channel operation in a WSN is to decrease the overall interference in the network, since the number of

nodes, which use same or adjacent channels will be decreased, when compared with single-channel networks. This can reduce packet corruptions and collisions. Another benefit is parallel communication. In single-channel sensor networks, the node pairs which are close to each other cannot communicate at the same time due to interference, however, in multi-channel networks, the close node pairs can communicate simultaneously, if they operate on non-overlapping channels.

ZigBee/802.15.4 standard is not the only short-range wireless technology that supports multiple channels in a radio. IEEE 802.11 (Wi-Fi) wireless technology also supports multiple channels. Both 802.15.4 and 802.11 can operate in 2.4 GHz ISM band. 802.11 can additionally operate in 5 GHz band. Even though IEEE 802.11b/g networks are widely deployed, IEEE 802.11n supporting higher data rates is becoming popular and widely used. Contrary to 16 channels in ZigBee, there are normally 14 channels in 2.4 GHz frequency domain (2.412 MHz, channel 1, to 2.484 MHz, channel 14) for Wi-Fi, where channels 1-13 are supported in most of the Europe and China, channels 1-11 are supported in United States and Canada, only channel 14 is supported in Japan and some countries have their own channel usage policy for Wi-Fi [11]. In most of the studies, only 11 channels (2.412 MHz to 2.462 MHz) are considered.

The multiple channels that can be available in Wi-Fi and ZigBee networks are shown in Figures 1.2 and 1.3, respectively. In Wi-Fi, as seen from Figure 1.2, there is a 5 MHz distance between each adjacent channel (except the channel 14) which is denoted as channel distance. One channel occupies 22 MHz (channel width), i.e., 11 MHz left and 11 MHz right from the center frequency. Since, 5 MHz is less than 22 MHz, adjacent channels overlap with each other, i.e., they interfere with each other . Since, $\lceil \frac{22}{5} \rceil = 5$, two channels x and y should be separated by at least 5 channels (i.e., $x - y$ should be ≥ 5) , in order to become non-overlapping (i.e., orthogonal). Although all channel pairs with at least 5 channel separation are orthogonal, usually the channels 1, 6 and 11 are referred as orthogonal channels, since these channels produce maximum number of non-overlapping channels, i.e., 3, in 11 channel Wi-Fi.

In ZigBee, as seen in Figure 1.3, there are 16 channels (2.405 MHz, channel 11, to 2480 MHz, channel 26) available in 2.4 GHz band. The channel numbers start from 11, because channel 0 is used for 868 MHz band, and channel 1 - 10 is used for 915 MHz [9]. Like Wi-Fi, there is a 5 MHz distance between each channel; this time, however, the channel width, which is 2 MHz, is less than the channel separation. Hence, all channels are orthogonal, and normally one should not expect any interference to occur between any two adjacent channels. However, [3] showed that there is a high interference between two adjacent channels, although they are theoretically orthogonal. In order to avoid this interference, 8 non-adjacent channels, i.e., only odd numbered or even numbered channels, are suggested to be used by [3].

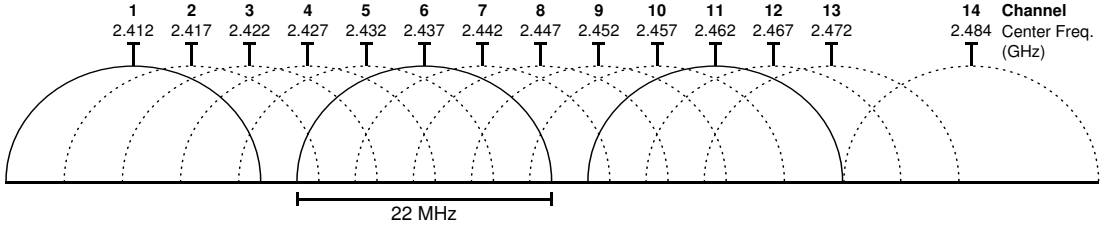

Figure 1.2: 2.4 GHz Wi-Fi channels [2].

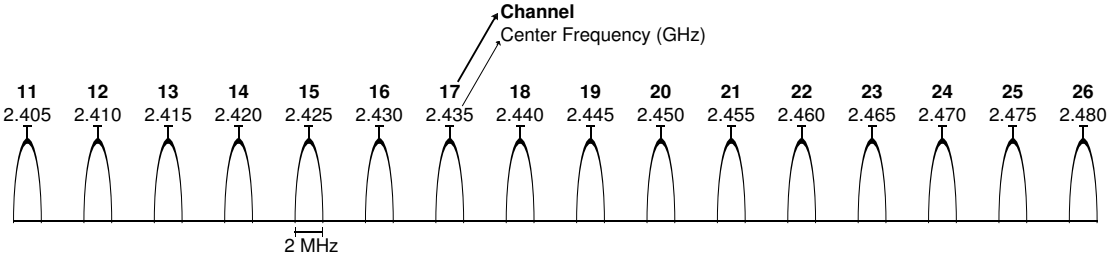

Figure 1.3: 2.4 GHz ZigBee channels (edited from [2]).

There are several sensor radio platforms supporting multi-channel operation. Table 1.1 gives a comparison of several 802.15.4 sensor radios: Nordic NrF905 [12], Chipcon CC1000 [13], Chipcon CC2420 [10], RFM TR1001 [14], Infineon TDA5250 [15]. The table also includes information about one Wi-Fi radio: Cisco HWIC-AP [16]. Some of the radios shown support multi-channel operation, as indicated in the table.

	Nordic NrF905	**Chipcon CC1000**	**Chipcon CC2420**	**RFM TR1001**	**Infineon TDA5250**	**Cisco HWIC-AP**
Operating Frequency	433/868/ 915 MHz	315/433/ 868/915 MHz	2.4 GHz	868.35 MHz	868 MHz	2.4 GHz
Modulation	GFSK	FSK	O-QPSK	ASK/OOK	ASK/FSK	BPSK, QPSK, 16-QAM, 64-QAM
Data Rate	50 kbps	76.8 kbps	250 kbps	115.2 kbps	64 kbps	54 Mbps
Max Tx. Power	10 dBm	10 dBm (433 MHz)	0 dBm	1.5 dBm	13 dBm	20 dBm
Receiver Sensitivity	-100 dBm	-107 dBm (868 MHz)	-94 dBm	-106 dBm	-109 dBm	-73 dBm (54 Mb/s)
Tx. Current	30 mA	26.7 mA	17.4 mA	12 mA	9 mA	-
Rx. Current	12.5 mA	9.6 mA	19.7 mA	3.8 mA	12 mA	-
Tx. Range (indoors)	50 m		50 m	100 m	30 m	24-90 m
Tx. Range (outdoors)	125 m	>100 m	125 m	300 m	80 m	90-610 m
Multi-Channel Support	+	-	+	-	-	+
No. of Channels	512	-	16	-	-	11
Channel Width	200 kHz	-	2 MHz	-	-	22 MHz

Table 1.1: Comparison of features of some 802.15.4 sensor radios and a Wi-Fi radio [4].

There are also node platforms that can support not only multiple channels, but also multiple radios. Such nodes are usually seen in wireless mesh networks, for which high throughput is extremely important [17, 18]. In mesh networks, nodes are much more powerful, which can be wireless access points, PCs or laptops, than the nodes in WSNs. In multi-radio networks, nodes can both transmit and receive at the same time, because they have more than one transceiver. Each transceiver can operate on a different channel. The radios can, even use different wireless technologies like IEEE 802.11 and 802.15.4 [19]. The radios of a node in a multi-radio wireless network, however, should be physically separated far enough from each other, in order to decrease the interference to each other. The distance between two radios should be more than 18 inches according to [20, 21] for radios that operate at the same frequency band (for example, at 2.4 GHz).

Although having multiple radios in nodes increases the overall throughput of the network, it needs more computing power, and consumes more energy compared to single radio networks. Additionally, they are more costly. Therefore, they are not preferred in WSNs. Additionally, the sensor nodes are usually very small and can not satisfy the distance requirement between the radios on the same node.

In literature, there are MAC protocols designed for multi-channel WSN communication to reduce interference [22, 23]. They usually use time synchronization for the communication of the nodes that do not operate on the same channel by switching their channels. However, time synchronization errors and channel switching delays affect the performance of limited power WSNs negatively as described in [3]. Hence, it may be preferable to assign a channel to a radio/node for an extended amount of time. This brings up the issue of how to assign channels to nodes in WSN in an appropriate manner. This problem is important even though the nodes in a WSN are using single radios, since a radio usually have multiple channels to choose from.

The channel assignment to nodes can be static or dynamic [24]. However, dynamic assignment may cause frequent switching of the channels for a node, which may require more resources to be consumed, and may also fail due to

clock drifts in nodes. Therefore, in this thesis we are focusing on static channel assignment, where channels of a WSN are assigned initially and stay like that for the whole duration of the network lifetime.

For static assignment, since the data is usually gathered from nodes to sink using routing trees, tree-based approaches are quite natural. In [3] authors propose a tree-based channel assignment protocol for WSNs, named GreedyPMIT, which does not switch the channels and do not use time synchronization among nodes. GreedyPMIT divides the network into trees rooted at BS and assigns each tree a unique channel, and forms the trees so that the overall interference of the network will be minimal. If the channels assigned to a tree are orthogonal with each other, there will not be any interference between the nodes of different trees. Then the only interference can occur between the nodes that belong to the same tree and that are close enough to interfere with each other. Due to the lack of inter-tree interference, the aim of the channel assignment is reducing the intra-tree interference in order to increase the overall throughput.

In this thesis, we evaluate and determine the weak points of GreedyPMIT and propose new WSN channel assignment algorithms to improve the performance. Our algorithms focus on the tree formation part of GreedyPMIT, and we propose new tree formation techniques and also a new interfere metric in order to increase network throughput. We evaluate the performance of our solutions with simulations, and show that our solutions perform better than GreedyPMIT with as much as 40% interference decrease.

The rest of this thesis is organized as follows: Chapter 2 includes the related work in the literature. In Chapter 3, we present the problem and our proposed algorithms. Simulation environment and the simulation results are described in Chapter 4, and finally the thesis is concluded in Chapter 5.

Chapter 2

Related Work

This section introduces some of the important multi-channel protocols in the literature in both wireless mesh/ad-hoc networks and WSNs.

2.1 Multi-Channel Wireless Communication

First we give some related work about multi-channel protocols in wireless networks, not specifically WSNs.

In [25], a MAC protocol for ad-hoc wireless networks, which utilizes multiple channels dynamically with the aim of increasing performance is proposed. IEEE 802.11 standard is used in the network for wireless communication, and they design a new MAC protocol rather than using the predefined 802.11 MAC, since 802.11 MAC is designed initially for single channel wireless networks. This predefined 802.11 MAC protocol does not perform well in multi-channel scenario, because of the multi-channel hidden terminal problem. Therefore, they propose a new protocol, which changes the channels of nodes dynamically to increase the network throughput.

In [26], Mishra et al. focuses on IEEE 802.11 interference, and defines two types of interference: co-channel (same channel) and adjacent (overlapping) channel interference. They also define an interference factor named I-factor, $I(i,j) = \frac{P_i}{P_j}$, that gives the fraction of a signal's power on channel j that will be received on channel i. P_i and P_j, which are the power received on channel i and j, are calculated when the receiver is operating on channel j. They state that use of non-overlapping channels in the network is not always necessary; overlapping channels can also be used.

[21] provides an algorithm for multi-channel wireless mesh networks (WMNs), where each node has multiple 802.11 network interface cards (multi-radio). Therefore, these nodes can listen more than one channel at a time. This paper states the central design issues as channel assignment and routing. For channel assignment, they propose channel switching mechanism, however, they do not switch the channels for each packet, they make it last for a longer duration, i.e., several minutes or hours. They divide the channel assignment problem into two parts: neighbor-to-interface binding and interface-to-channel binding, where neighbor-to-interface binding decides which interface is used for communicating with neighbor nodes and interface-to-channel binding decides which channel is used for the interface, i.e., radio. And they present a distributed routing and channel assigning algorithm that utilizes the local topology and local traffic load.

2.2 Multi-Channel Wireless Sensor Networks

WSNs are special type of wireless networks, where there are a lot of constraints on the computational power, data rate, packet size, RAM, storage and power of the nodes. Since WSNs are different, and usually less powerful than wireless adhoc or mesh networks, they require different protocols. This section introduces some of the works done in multi-channel WSNs in literature.

In [22], Zhou et al. propose a channel assignment scheme, and divide their protocol into two parts: frequency assignment and media access. For frequency assignment, in order to reduce the interference and hidden terminal problems, nodes within two communication hops are assigned with different frequencies, if possible. In media access, neighbor nodes can compete for the medium by using a slotted CSMA protocol. For frequency/channel assignment, they propose four different solutions, and they compare their frequency assignment and media access protocol with CSMA, and show that their protocol performs better in aggregate throughput, packet delivery ratio, energy consumption per delivered data byte. It is also stated that, although there are lots of multi-channel MAC protocols proposed for wireless ad-hoc networks, these protocols are not suitable for WSNs. The packet sizes in WSNs are relatively small, therefore, RTS/CTS packets used in wireless ad-hoc networks are an overhead, and not suitable in WSNs.

In [23], Zhang et al. propose a multi-channel MAC protocol, which is based on Time Division Multiple Access (TDMA). This algorithm requires time synchronization for nodes, in order to switch the channels to a common frequency for communicating pairs. This protocol also supports both unicast and broadcast communication.

In [3], Wu et al. propose a tree-based multi-channel protocol, called Greedy-PMIT, for WSNs to decrease the interference in the network. Like other papers in literature, they divide the problem into two main parts: channel assignment and routing. However, their proposed solution forms a shortest-path tree, which eliminates the complexity of the routing algorithm, because the routing is as simple as forwarding packets to the parent of each node. They propose a solution, which partitions the network into k trees, where k is the number of available non-overlapping channels. By assigning non-overlapping channels to each tree, they eliminate the effect of inter-tree interference, and the only problem remains is the intra-tree interference, which is the interference between each node in the same tree.

Wu et al. state and show that time synchronization between each node and channel switching brings an overhead to WSNs, because of their limited power processors and clock drifts. Therefore, they avoid using time synchronization and channel switching in their algorithm, and they focus on partitioning the network into trees. With k available channels, they partition the network into k vertex-disjoint trees with the goal of minimizing the maximum intra-tree interference, which is measured by their proposed interference metric. Their algorithm uses a greedy approach in selecting channels. At the end of the execution of the algorithm, the network is divided into k trees rooted at the base station (BS), which is assumed to have k radio transceivers operating on k channels. In this way, at least k parallel communications are supported in the network. Since a tree-based approach is used, the routing of sensor data to the sink node (BS) is very easy.

Jeong et al. [19] propose a WSN architecture, where network is divided into clusters, and the cluster-heads have dual heterogeneous radios. Sensor nodes except the cluster-heads have only IEEE 802.15.4 radios. One of the dual radios in a cluster-head is used to communicate with sensor nodes, and the other radio, which is a 802.11 radio, is used to communicate with other cluster-heads. Cluster-heads have more processing power and more powerful battery in order to increase the life-time of the network.

In the paper, they complain about the interference between 802.11 and 802.15.4 radio interfaces in cluster-heads. Although the channels used by 802.11 and 802.15.4 radios are chosen as orthogonal, especially 802.11 traffic causes severe interference on 802.15.4 traffic. They state that even 30 cm distance between the radios do not cancel the interference. They propose an adaptive aggregation and scheduling algorithm for the cluster-heads in order to decrease the interference.

Chapter 3

Our Proposed Channel Assignment Schemes

There are multi-channel MAC protocols like [22, 23] proposed in order to improve the network performance, in which the sensor nodes are time synchronized to provide communication between nodes. In [22], nodes within two hops are tried to operate on different channels to decrease the interference, which is not always feasible for networks that have few available channels. Although these proposed protocols improve the network performance compared to single-channel protocols, these algorithms are not very effective for WSNs. Due the computing power limitations and clock drifts, channel switching and time synchronization are significant overheads in WSNs, as described in [3].

An algorithm with static channel assignment or reduced channel switching will be more accurate for WSNs. Also another problem for a WSN can be calculating the routing paths, which also should not be found with an expensive routing algorithm, because of the constraints of the sensor nodes. In order to propose a light-weight practical protocol for WSNs, we should avoid time synchronized protocols and frequent channel switchings, and also choose a simple routing algorithm. Therefore, we choose GreedyPMIT [3], as our base protocol and approach, which assigns channels to sensor nodes in a tree-based scheme

without time synchronization and channel switching. It also uses a very simple routing procedure. In this thesis, we first identify some drawbacks of Greedy-PMIT, and then we propose new algorithms with a similar approach in order to increase the network performance.

In this chapter, we first introduce GreedyPMIT algorithm and describe some weak points of that. Then we introduce our proposed methods and describe them in detail.

3.1 GreedyPMIT Algorithm Proposed in the Literature

In [3], the authors propose a Tree-based Multi-Channel Protocol (TMCP) for channel assignment in WSNs that increases WSN throughput. To achieve that, they divide the network into k vertex-disjoint trees rooted at the sink node, i.e., base station (BS), where k is the number of available channels; k can be, for example, 8 or 16 for 802.15.4 networks. Each tree operates on a different non-overlapping channel, so that there will not be any interference between any two trees, i.e., zero inter-tree interference. Then, the only interference source will be the interference between same tree nodes, i.e., intra-tree interference. Therefore, the aim of TMCP is to minimize the intra-tree interference.

To minimize the intra-tree interference, they first define an interference metric. They consider the interference of a node as the interference that can be potentially received by the node from the other same-tree nodes in its interference disk. Interference disk is a circle centered at the node of interest and having a radius equal to the interference range. Interference range of a node is generally larger than communication range of the node. It is usually considered to be 1.5 or 2 times of the communication range. After calculating the interference values of all nodes, the interference value of a tree T is defined as the interference of the non-leaf node with maximum interference value, i.e., $int(T) = max\{int(u)$: u is a non-leaf node of $T\}$. They do not take the interference value of the leaf nodes

Algorithm 1 GreedyPMIT in [3]

Input: k channels, a graph $G = (V, E)$, a root r, and the interference set of every node.

Output: For each node u, $channel_u$ and $parent_u$.

use BFSFatTree algorithm to construct a fat-tree rooted at r.
for each channel i **do**
  $T_i = r$;
end for
for each node u **do**
  $channel_u = 0$;
  $parent_u = null$;
end for
$level = 1$;
repeat
  $node\_list = \{u | height(u) == level;\ channel_u == 0\}$.
  sort $node\_list$ in ascending order by the number of node's parents.
  for each node u in $node\_list$ **do**
    find T_i which keep connected and has the least interference after adding u.
    $T_i = T_i \cup \{u\}$;
    $channel_u = i$;
    $parent_u = v$, which connects u and has the least interference among all nodes in T_i.
    update the interference value of T_i.
  end for
  $level$++;
until $level >$ the maximum height of the fat tree

Algorithm 2 BFSFatTree in [3]

Input: a graph $G = (V, E)$ and a root r.
Output: For every node u, its parent set *parentSet(u)* and its height in the tree *height(u)*.

```
1: for each node u in G do
2:    height(u) = MAXIMUM INTEGER;
3:    parentSet(u) = null;
4: end for
5: S = r;
6: height(r) = 0;
7: for each node u in S do
8:    for each node u's neighbor v do
9:       if height(v) > height(u) then
10:         height(v) = height(u) + 1;
11:         parentSet(v) = parentSet(v) ∪ {u}.
12:         S = S ∪ {v}.
13:      end if
14:   end for
15: end for
```

into account, since leaf nodes in a WSN do not receive data.

The aim is to partition the network so that the maximum intra-tree interference of all trees will be minimum. In [3], they state that this problem is called PMIT problem and prove that it is NP-Complete. Consequently, they propose a greedy algorithm, named GreedyPMIT, that tries to minimize the maximum intra-tree interference value of all the trees. They cannot find the exact minimum interference, however, they decrease the interference when compared to single channel communication and a multi-channel Eavesdropping channel assignment method proposed in [22].

Algorithm 1 explains GreedyPMIT Algorithm, where they first apply a Breadth-First Search (BFS) algorithm to form a fat tree rooted at the BS (Algorithm 2). That fat tree is a shortest path tree. Nodes keep their heights in the tree, and they may have multiple parents. From the first level of the fat tree (children of the BS) to the last level, the algorithm assigns channels to the nodes one-by-one for each level. For each level, the nodes in that level are sorted in ascending order by the number of parents in the fat tree. Then in that sorted

order, the algorithm assigns a channel, i.e., tree to that node, which has the least interference after adding that node. They also prove that the complexity of their algorithm is $O(dkn^2)$ in worst case, where d is the diameter of the graph $G = (V, E)$, where V is the set of nodes, and E is the set of edges indicating direct reachability, n is the number of nodes, and k is the number of channels.

After some examination of that algorithm and implementation of it in Java, we find some weak points that can be improved. Once GreedyPMIT algorithm assigns a channel to one node, there is no turning back; this is the consequence of a greedy algorithm. Therefore, assigning one channel to a node may seem to be better for that level, however, this can lead to a worse choice at further levels. This is inevitable for greedy algorithms, and choosing a greedy algorithm for this problem is not a mistake, since finding the exact minimum interference is an NP-Complete problem. Therefore, we try to improve the algorithm to get closer to the minimum interference.

3.2 Proposed Solution

First, we define an additional interference metric similar to the metric in [3]. Our metric is also taking the physical distance between nodes in their interference disk in the same tree into account rather than just the number of nodes. In order to calculate the distance between nodes, we should know the locations of the sensor nodes. These locations can be found with GPS modules, or coordinates of the nodes can be kept in a database. This new interference value of a node is the sum of $\frac{1}{d^2}$ of each same-tree node in its interference disk, which are in the same tree with that node (d: distance to the node). In a formal way, the interference value of a node u is calculated as $\sum_{i=1}^{|N|} \frac{1}{d(u,v_i)^2}$, where N is the set of same-tree nodes in the interference range of node u, $v_i \in N$, and $d(u, v_i)$ is the distance between nodes u and v_i. After calculating the interference values of all nodes, the interference value of a tree T is defined again as the interference of the non-leaf node with maximum interference value, i.e., $int(T) = max\{int(u)$: u is a non-leaf node of $T\}$. We define this additional metric, since not only the number of nodes

in the interference disk, but also the distance between the nodes are important for measuring the interference. We also want to examine the effect of this metric.

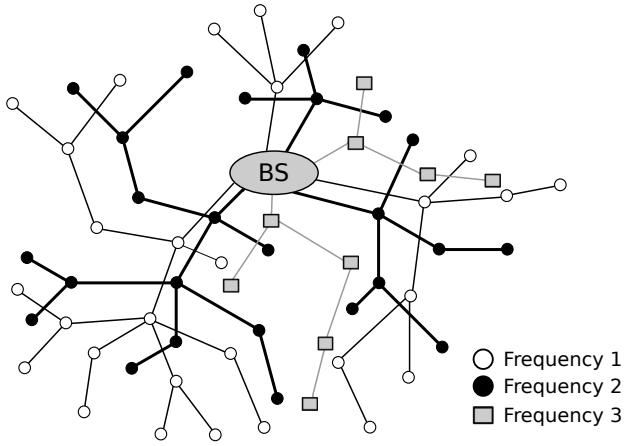

Figure 3.1: Example tree formation in [3].

GreedyPMIT assigns channels to the nodes from the first level of the tree to the last level, i.e., top-down approach. Expected tree formations of GreedyPMIT is shown in [3] with Figure 3.1. As seen from the figure, they do not want blocks of nodes that are all in the same tree; their aim is to equally distribute the nodes into trees in the area. Although our implementation of GreedyPMIT shows tree formations as described in that figure, there can be some bad tree formations as well, like in Figure 3.2, which is drawn by our Java implementation. In that figure, we have a network with 121 nodes, default interference metric of GreedyPMIT is used, the communication range and the interference range are set as 1.5 and 2.25 units, respectively. The network is assumed to be a perfect grid. When perfect grid topology is considered, 1 unit is the distance between a node's left, right, top and bottom neighbors, and 1.5 units communication range contains at most 8 neighbor nodes in a node's vicinity, which can be seen from the inner dashed circle centered at BS (middle node with the plus sign) in Figure 3.2. The interference range is set as 2.25 because it is set to 1.5 times of the communication range. Interference range of 2.25 units contains 20 neighbors, as seen from the outer dashed circle in the same figure. The number of channels are set to 3 in this example, that can be understood from the number of different shapes. The tree formation in the figure is not formed as described in Figure 3.1 due to the greedy property of the algorithm. Blocks of trees are formed unintentionally (i.e.,

same tree nodes are clustered in a region).

In the top-right of the figure, there is a block of square-shaped tree, in the left-top there is a triangle-shaped tree block, and there are circle-shaped tree blocks in the sides. These blocks increase the interference of the network because all or most of the nodes in a node's interference range become in the same tree, which increases the interference value of the network. In the example in Figure 3.2, the interference values for square, triangle and circle-shaped trees are 16, 15 and 13, respectively. The maximum of these values are taken as the interference of the network, which is 16.

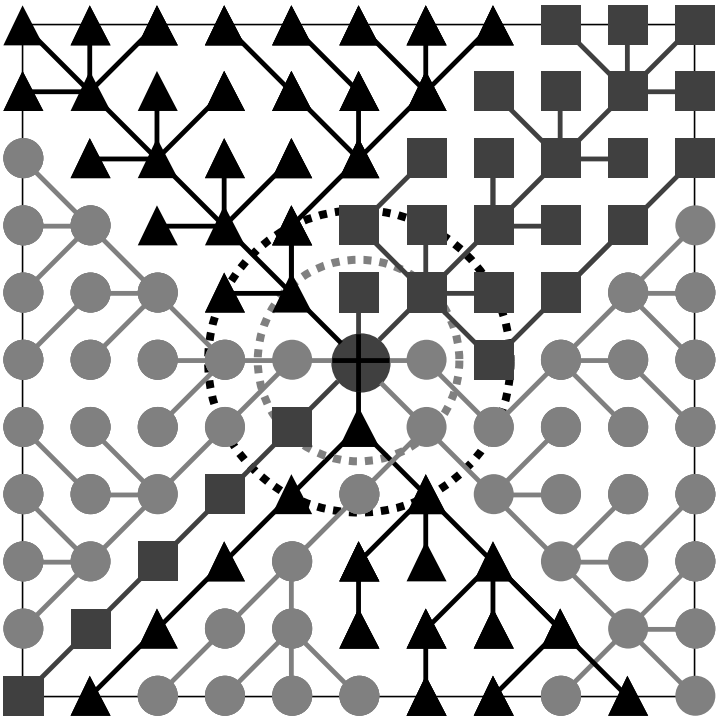

Figure 3.2: Example tree formation when GreedyPMIT-N is used.

Our main aim is to decrease the number of blocks, which will also help to decrease the interference of the network. To overcome this problem, we define two new algorithms which are based on GreedyPMIT. These two algorithms have two versions each, where first version has the default interference metric, i.e., number of nodes in the interference disk, and the second one has the distance-based interference metric as explained before. We refer to the original GreedyPMIT with node-count based interference metric as GreedyPMIT-N (N comes from

Node-count) and GreedyPMIT with new interference metric, i.e., distance-based interference metric as GreedyPMIT-D (D comes from Distance). Then we name the algorithms so that the algorithms which have suffixes -N and -D are the algorithms with default (number of nodes) and distance-based interference metric, respectively. Then our two new algorithms are named as Bottom Up Channel Assignment-N (BUCA-N) and Neighbor Count based Channel Assignment-N (NCCA-N), and their other versions with distance-based interference metric are named as BUCA-D and NCCA-D, respectively.

Since GreedyPMIT can form inefficient trees like Figure 3.2, we want to decrease the negative effects of this greedy algorithm. Therefore, we want to form initially more than k trees and unite those trees to k trees intelligently, in order to decrease the negative effects of the greedy algorithm. By doing this, we try to decrease the sizes of blocks of trees, which increase the interference. The number of initial trees can be as much as the number of neighbors of BS, therefore we form initially c trees.

BUCA-N and BUCA-D form the trees from the bottom of the tree to the top (level 1) of it, as described in Algorithms 3 and 4. In Algorithm 3, like GreedyPMIT, first a BFS fat tree is formed using Algorithm 2. Unlike GreedyPMIT, we start from the last level of the fat tree for assigning channels to the nodes. For each level from bottom to top, the nodes in that level are sorted in ascending order by the number of parents in the fat tree. Then in that sorted order, the algorithm assigns a channel, i.e., tree to that node, which has the least interference after adding that node. We also take the distance between a node and its parent, and the number of children of a parent into account in BUCA-N and BUCA-D.

NCCA-N and NCCA-D form the trees in the same way with GreedyPMIT (top-down), however, they consider initially c number of trees, where c is the number of children BS has. In fact, all our four algorithms behave like that: forming c (child count of BS) number of initial trees, instead of k (channel count) number of initial trees, and then they decrease the number of trees to k by uniting c number of initial trees. Therefore, for NCCA, we give c as the channel number input to GreedyPMIT (Algorithm 1). Also after the last line of GreedyPMIT, line

Algorithm 3 BUCA

Input: k channels, a root r, a graph $G = (V, E)$, and the interference set of every node.

Output: set of c initial trees T, where c is the number of neighbors of root r, $channel_u$ and $parent_u$ for every node u.

use BFSFatTree algorithm described in [3] to construct a fat-tree rooted at r.
for each node u **do**
  $channel_u = 0$; $parent_u = null$; *addedChildrenSize(u)* $= 0$;
  childrenSize(u) $=$ # of children count in BFS Fat tree;
end for
$c = 0$;
for each neighbor n of r **do**
  $T_c = \{r, n\}$; $channel_n = c$; $parent_n = r$; c++;
end for
level $=$ maximum level of BFS Fat tree;
repeat
  node_list $= \{u | height(u) == level;\ channel_u == 0\}$.
  sort *node_list* in ascending order by the number of node's parents.
  for each node u in *node_list* **do**
    parents $= \{p | p \in parentSet(u)\}$.
    onlyChild $= \{p | p \in parents;\ childrenSize(p) == 1\}$.
    if *onlyChild* $\neq \emptyset$ **then**
      farthestParent $= \{p | p \in onlyChild;\ distance(u,\ p) == max\}$.
      AddChild(*u, farthestParent*[*random available index*]).
    else
      noAddedChildren $= \{p | p \in parents;\ addedChildrenSize(p) == 0\}$.
      if *noAddedChildren* $\neq \emptyset$ **then**
        minChild $= \{p | p \in noAddedChildren;\ childrenSize(p) == min\}$.
      else
        minInterf $= \{p | p \in parents;\ interf(p,u) == min\}$.
        minChild $= \{p | p \in$ minInterf; childrenSize(p) $==$ min$\}$.
      end if
      farthestParent $= \{p | p \in minChild;\ distance(u,p) == max\}$.
      AddChild(*u, farthestParent*[*random available index*]).
    end if
  end for
  level$--$;
until *level* > 1
calculate *unionInterf* matrix, where *unionInterf*(i,j) denotes the interference value of the tree i, when it is united with tree j.
UniteTrees(*unionInterf, k, c, T*).

Algorithm 4 AddChild

Input: child u and a parent p.
Output: T_i, $channel_u$, $parent_u$, *addedChildrenSize(p)*.

```
1: if DummyT_u == ∅ then
2:     DummyT_u = {u};
3: end if
4: if DummyT_p == ∅ then
5:     DummyT_p = {p};
6: end if
7: if p is not a level 1 node then
8:     DummyT_p = DummyT_p ∪ DummyT_u;
9: else
10:    find T_i where p ∈ T_i.
11:    T_i = T_i ∪ DummyT_u;
12:    channel_u = i;
13: end if
14: parent_u = p;
15: addedChildrenSize(p)++;
```

21, we should calculate *unionInterf* matrix and call *UniteTrees* Algorithm similar with BUCA. Table 3.1 summarizes the differences between our four algorithms and GreedyPMIT.

Algorithms	**Initial Tree Formation**	**No. of Trees Formed Initially**	**Unites Initial Trees**	**Interference Metric**
GreedyPMIT-N	Top-down	k	No	No. of Nodes
GreedyPMIT-D	Top-down	k	No	Distance Between Nodes
NCCA-N	Top-down	c	If (k < c)	No. of Nodes
NCCA-D	Top-down	c	If (k < c)	Distance Between Nodes
BUCA-N	Bottom-up	c	If (k < c)	No. of Nodes
BUCA-D	Bottom-up	c	If (k < c)	Distance Between Nodes

Table 3.1: Comparison of our proposed algorithms and GreedyPMIT.

Then for all our algorithms, we consider the available number of channels, k, and if it is less than c, we unite those initially formed trees so that the final number of trees will be equal to k. If k is greater than or equal to c, there will not be any union operation, and if k is greater than c, as same with GreedyPMIT,

$k - c$ channels cannot be used, since BS can communicate with only c nodes. The union operation tries to minimize the maximum interference. The union operation is same in all our algorithms, the difference between each algorithm is the forming procedure of the initial c number of trees.

Algorithm 6 explains the union operation. At first we have c trees, and we try to decrease them to k trees. To do this we propose an iterative approach, which decreases the number of trees step by step by pairing, i.e., uniting the trees in each iteration. Before pairing, we calculate the number of pairs needed for each iteration as explained in Algorithm 5. We want to unite the initial c number of unpaired trees as $N_1, N_2, \ldots, N_k$ assuming $|N_i - N_j| \leq 1$ for any i, j to form a balanced network, where N_i denotes the number of initial trees paired, i.e., united to form the final tree T_i, and $\sum_{i=1}^{k} N_i = c$. Although, we calculate the number of pairs according to the assumption before in *CalculatePairs* method, this assumption is needed for calculating the number of pairs, and the final tree may not ensure this assumption when uniting the residual trees in Algorithm 6. We calculate the number of pairs in an iteration by calculating the closest even numbers less than or equal to all N_is, summing them, and dividing the result of the sum by 2, as calculated in Algorithm 5 and shown in Figure 3.5. As a remark, calculated pairs can be at most the half of the unpaired trees for each iteration, since the summation described before can be at most the number of unpaired trees, i.e., *upt*, which is a parameter of *CalculatePairs* method.

We pair the trees as described in Algorithm 7 so that the interference is reduced, and move to the next iteration with less and bigger trees. This algorithm takes *unionInterf* matrix, where *unionInterf*(i,j) denotes the interference value of the tree i, when it is united with tree j, as an input. We mark the elements in this matrix increasingly, until we can pair p pairs, where p denotes the number of pairs needed. We repeat this procedure in Algorithm 6, until we reach k trees, i.e., $neededPairs = 0$. For some iterations, we have some residual trees, which are not paired. At the end, in reverse order, i.e., from the last iteration to the first, we add these trees to one of the k trees formed before, in each iteration, as explained in Algorithm 6, which tries to minimize the global maximum interference. Finally, we will have k trees with hopefully less interference.

Algorithm 5 CalculatePairs

Input: number of unpaired trees *upt*, number of available channels k.
Output: *neededPairs*.

minBranchedTree = $\lfloor \frac{upt}{k} \rfloor$.
remainedBranches = *upt* % k;
if *minBranchedTree* % 2 == 1 **then**
    maximumEvenNo = (*minBranchedTree* -1) $\times k$ + *remainedBranches* $\times$ 2;
else
    maximumEvenNo = *minBranchedTree* $\times k$;
end if
neededPairs = *maximumEvenNo* / 2;

Algorithm 6 UniteTrees

Input: *unionInterf* matrix, where *unionInterf*(i,j) denotes the interference value of the tree i, when it is united with tree j, k number of available channels, c number of neighbors of BS, and set of trees T.
Output: final tree T.

if $k < c$ **then**
    neededPairs = *CalculatePairs(c, k)*;
    level = 0;
    while *neededPairs* > *0* **do**
        T = *MarkAndPair(T, unionInterf, neededPairs, level)*;
        neededPairs = *CalculatePairs(neededPairs, k)*;
        level++;
    end while
    if $\lfloor \frac{c}{k} \rfloor \geq 2$ **then**
        for i = *level* $\rightarrow$ 0 **do**
            for each tree T_i in *residuals*[i] **do**
                unite T_i with a tree in T which has the least interference after adding T_i.
            end for
        end for
    end if
end if

Algorithm 7 MarkAndPair

Input: set of trees T, *unionInterf* matrix, where *unionInterf*(i,j) denotes the interference value of the tree i, when it is united with tree j, p denotes the number of pairs needed, and level l.

Output: paired trees T, residual trees *residuals*.

```
1: maxPairs = 0;
2: max = 0;
3: count = 0;
4: while maxPairs < p do
5:    set every element of mins array with MAXIMUM INTEGER.
6:    for i = 0 → T.length do
7:       for j = 0 → T.length do
8:          if i ≠ j and unionInterf[i][j] > max and unionInterf[i][j] < mins[i]
            then
9:             mins[i] = unionInterf[i][j];
10:         end if
11:      end for
12:   end for
13:   sort the mins array in ascending order.
14:   if count == 0 then
15:      max = mins[2 × p − 1];
16:   else
17:      max = mins[0];
18:   end if
19:   for i = 0 → T.length do
20:      for j = 0 → T.length do
21:         if i ≠ j and unionInterf[i][j] ≤ max then
22:            marked[i][j] = true;
23:         end if
24:      end for
25:   end for
26:   maxPairs = number of the trees in maximum matching calculated by using
      marked elements.
27:   count++;
28: end while
29: unite the trees according to maximum matching.
30: residuals[l] = set of unmatched trees in this level l (if any).
```

In Figure 3.2, we showed an inefficient example of GreedyPMIT, where it produces blocks of nodes, which are the elements of the same tree. We present our algorithms, which try to overcome this problem, and hopefully produce better results. In Figure 3.3, we can see the same network with same configuration when BUCA-N is used. We can observe that the trees are distributed in the network more uniform than Figure 3.2, and it produces a similar network presented in Figure 3.1, which is used in [3] as an example. In this solution, the interference values for square, triangle and circle-shaped trees are 11, 9 and 9, respectively. As a result, the interference value of the network is 11, which has lower interference value than GreedyPMIT-N, 16, for this network configuration.

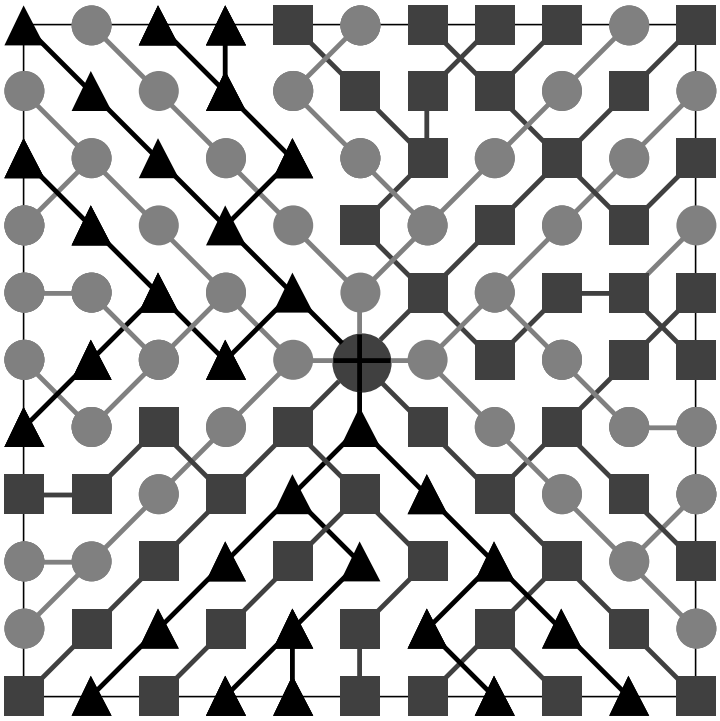

Figure 3.3: Example tree formation when BUCA-N is used.

The representation of the same network configuration solved with NCCA-N is shown in Figure 3.4, where it has more blocks than BUCA-N and less than GreedyPMIT-N. In this network, the interference values for square, triangle and circle-shaped trees are 12, 9 and 13, respectively, with the maximum of 13. Again this algorithm performs better than GreedyPMIT-N in this example. To compare our algorithms with GreedyPMIT, we perform simulations with different

scenarios, which are explained in detail in the following chapter.

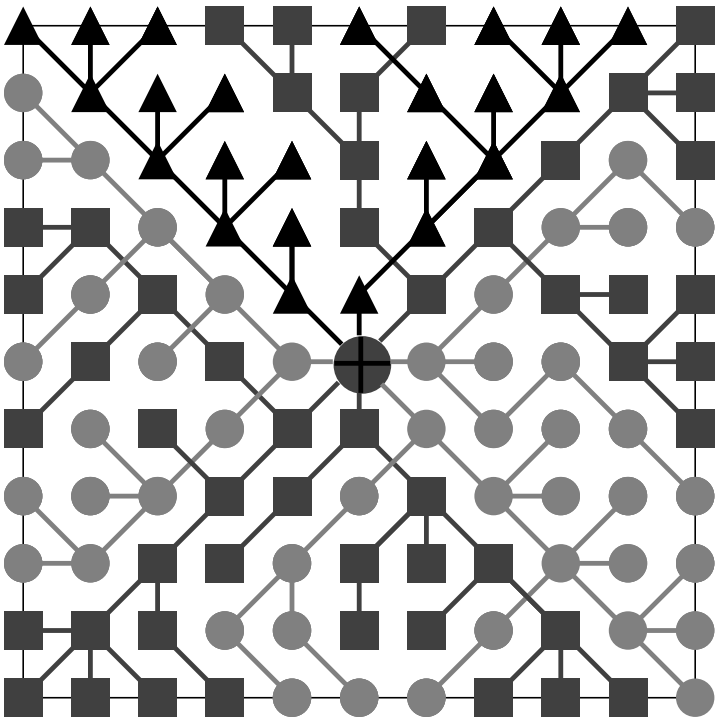

Figure 3.4: Example tree formation when NCCA-N is used.

3.2.1 Uniting Trees

Figure 3.5 shows an example of the union operation, where $c = 16$ and $k = 3$. Although maximum value of c can be at most 12 in the simulations, we choose a larger number to examine the union operation better. At the first step, we calculate $\lfloor \frac{c}{k} \rfloor$ as $\lfloor \frac{16}{3} \rfloor = 5$, and try to divide the 3 trees as union of 5 initial (Step 1) trees, i.e., 5 - 5 - 5. Then we should add the trees remaining, which is 16 % 3 = 1 (only one tree), consequently our 3 trees will be united as 6 - 5 - 5 number of initial trees. This is the final version of the trees. We unite the trees so that, the difference between number of united trees for each final tree is minimum, i.e., 6 - 5 = 1. However, these numbers are used only for calculating the number of pairs for each step, and at the end, the trees can be formed as 7 - 5 - 4 or another combination if this gives the minimum interference. However, in this figure we assume that the final tree is a 6 - 5 - 5 tree. The next thing for this iteration is to

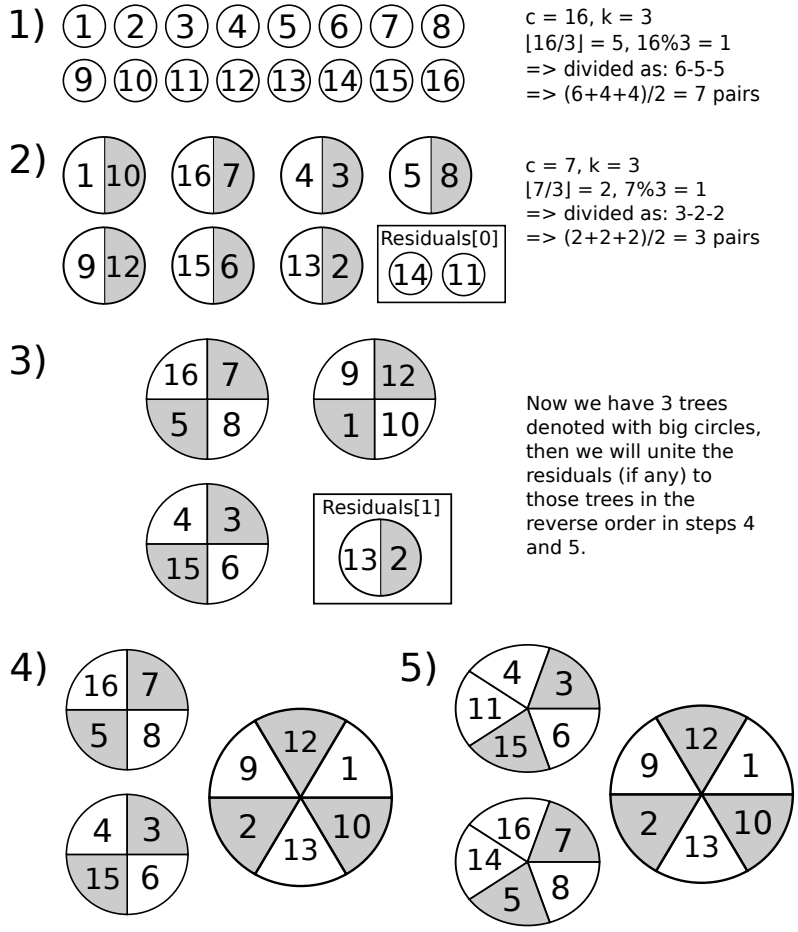

Figure 3.5: Demonstration of uniting trees.

calculate the number of pairs needed, which is the greatest even number for each tree, which is less than or equal to the united trees count. Therefore, we take 4 for the trees with united tree counts equal to 5, and take 6 for the other one. Then we add those even numbers, and divide by 2 to find the number of pairs. By $\frac{6+4+4}{2}$, we get 7 pairs for this iteration as described in Algorithm 5. Finally we should pair these 7 trees so that the interference will be minimum. The pairing operation is explained in Algorithm 7. We can assume that the pairing operations in Figure 3.5 are done according to that algorithm.

In the second step, the trees in the previous step is paired according to Algorithm 7, and we get 7 paired trees and 2 residual trees. The residual trees are the trees that do not have any pair in this level. This two trees come from the

rounding of two 5's to 4 in the previous step, and they left unpaired in order to use in the following steps. Similar to the previous step, we calculate the number of pairs with a result of 3.

At the third step, we get $k = 3$ trees, where each of them are the union of 4 trees. Also we get a residual tree in this level, and we save it to use later. In this step, we reach k trees, therefore, we do not need to calculate the pairs in the following step, which can be calculated as 0 from Algorithm 5. What we need to do is to unite the residual trees to form the final tree. When uniting the residual trees, we unite them in the reverse order, i.e., start to unite the Step 3's residual tree in the fourth step, and the Step 2's residual trees in the last step. Finally, we have 3 trees that are the union of 16 trees, and the 6 - 5 - 5 formation can be seen from the united trees, where 2 trees are the union of 5 Step 1 (initial) trees, and the other is the union of 6 Step 1 trees.

3.2.2 Marking and Pairing Trees

Pairing operation is described in Algorithm 7, in this section, we present a visual description for marking and pairing the trees. To determine which tree will be paired with which one, at first we should calculate the new interference values when tree i is paired (united) with tree j, for all (i,j) pairs. Then we form a matrix M, where the value in $M(i, j)$ is equal to $M(j, i)$, and it is the interference value when tree i is united with tree j. And also we label $M(i, i)$ as X, which is not processed in any step in the algorithm. Procedure of choosing the pair is described in Tables 3.2, 3.3 and 3.4 with an example. In this example, we have 7 trees to be united, and the number of pairs needed to be formed in this iteration is 3. Therefore, at the end, six trees will have their unique pairs, and the other tree will be a residual tree.

To pair the trees with minimal interference, we develop a straight forward method. First of all, we find the minimum interference values in each row, and sort them in ascending order. In the example, the minimum numbers are 13, 13, 14, 16, 15, 17 and 14 in the row order. Then, ascending order sorted version of

	1	2	3	4	5	6	7
1	X	**13**	**14**	19	**15**	21	**14**
2	**13**	X	26	**16**	18	17	19
3	**14**	26	X	18	22	32	25
4	19	**16**	18	X	23	19	20
5	**15**	18	22	23	X	40	33
6	21	17	32	19	40	X	25
7	**14**	19	25	20	33	25	X

Table 3.2: Demonstration of marking, step 1.

them are 13, 13, 14, 14, 15, 16, 17. Since we need 3 pairs, we first select the first $3 \times 2 = 6^{th}$ minimum number, and check if we can pair 3 trees by using it. To do this, we set the maximum interference max as 6^{th} minimum number, that is 16. The second step is to mark the elements of the matrix, which are less than or equal to $max = 16$ as seen from bold elements in Table 3.2. Later on, we check if we can form 3 pairs with all unique elements with marked elements. By all unique, we mean that if a tree i is paired with a tree j, then it cannot be paired with any tree again, therefore, the remaining pairs cannot contain tree i or j.

We check how many trees can be paired with given marked trees with a brute force approach in the tests with $O((r-1)!)$ complexity in worst case, where r is the number of rows in the matrix. This pairing problem is defined as finding the maximum matching in a non-bipartite graph. In the graph $G = (V, E)$, V is the set of trees, i.e., 7 trees in the example, and E is the set of edges between them, where if $M(i, j)$ is marked, there is an edge between i and j. Matching is the set of edges in the graph, which contains unique vertices. Maximum matching is a matching, which has maximum number of edges possible. This definition is same with our maximum pair definition, where we calculate the maximum pairs in the given marked trees, and compare the result with the number of pairs needed. If we reach the number of pairs needed, we pair the trees according to the maximum matching. Although our solution in our simulations have a $O((r-1)!)$ running time, the running time can be decreased to $O(|V|^4)$ by using Edmond's Maximum Matching Algorithm [27] or $O(|E|\sqrt{|V|})$ by using Micali and Vazirani's Maximum Matching Algorithm [28].

	1	2	3	4	5	6	7
1	X	**13**	**14**	19	**15**	21	**14**
2	**13**	X	26	**16**	18	**17**	19
3	**14**	26	X	18	22	32	25
4	19	**16**	18	X	23	19	20
5	**15**	18	22	23	X	40	33
6	21	**17**	32	19	40	X	25
7	**14**	19	25	20	33	25	X

Table 3.3: Demonstration of marking, step 2.

By using marked trees, we calculate the maximum number of pairs using a maximum matching algorithm. For example, the maximum number of pairs that can be formed according to Table 3.2 is 2. One example of these 2 pairs are {1,3} and {2,4}. As seen from the table, if {1,2} was chosen as a pair, we could not find any second pair, therefore, either 3, 5 or 7 should be chosen as 1's pair to find a maximum matching in this step.

The iteration described will be repeated until the number of maximum matching is greater than or equal to the number of pairs needed. In the next iteration, we again find the minimum interference values in each row, however, we skip the marked interference values, and do not take them into account while finding the minimum. Then in these r minimum interference values, we find the minimum of them, and set as *max*. Then we mark all the elements in the matrix, which are equal to *max*. In the example, *max* is equal to 17, which is the minimum number in all unmarked elements. Then we mark all elements in the matrix that are equal to 17 as seen from Table 3.3. Next, we calculate the maximum matching with given marked trees. Only the pair {2,6} is added to the previous pair list and this does not help to form the third pair, since as we mentioned before tree 2 should be paired with tree 4 in order to form 2 pairs, therefore, we cannot use the newly added pair {2,6}, and cannot form 3 pairs in this iteration.

Now we need to find the next minimum unmarked interference value, and calculate the maximum matching. The next minimum unmarked interference value is 18, then we set *max* as 18, and mark the values which are equal to

	1	2	3	4	5	6	7
1	X	**13**	**14**	19	**15**	21	**14**
2	**13**	X	26	**16**	**18**	**17**	19
3	**14**	26	X	**18**	22	32	25
4	19	**16**	**18**	X	23	19	20
5	**15**	**18**	22	23	X	40	33
6	21	**17**	32	19	40	X	25
7	**14**	19	25	20	33	25	X

Table 3.4: Demonstration of marking, step 3.

18. The new matrix in this iteration can be seen from Table 3.4. And now, we can have 3 pairs with these marked trees, and one example of the maximum matchings are {1,7}, {2,5} and {3,4}, with residual tree 6. As you can see from the table, another maximum matching can be formed like {1,5}, {2,6} and {3,4} with residual tree 7. And our algorithm chooses one of the maximum matchings according to the algorithm used for maximum matching.

3.2.3 Complexity

Due to the fact that PMIT problem is NP-Complete, the proposed algorithm in [3] is greedy, and has a time complexity of $O(dkn^2)$ in worst case, as explained before. Therefore, our algorithm should also have a polynomial time complexity. The time complexity of GreedyPMIT comes from the tree formation part, however, in our case the complexity of the union of the trees should also be considered. We have a similar time complexity in the initial tree formation part with GreedyPMIT, however, rather than forming k trees we form c trees.

As seen from Algorithm 6, we repeat calling *MarkAndPair* and *CalculatePairs* methods until the *neededPairs* are greater than 0. In each iteration, *neededPairs* decreases by pairing two trees, and we put the residual trees (if any) into *residuals* array as described in Figure 3.5. Since we are dealing only with the paired trees, not the residuals in the while loop, the number of iterations in that while loop depend on only the paired trees, and *neededPairs*, which is calculated according to

the paired trees. In the worst case *neededPairs* can be decreased into $\lfloor \frac{neededPairs}{2} \rfloor$ in each iteration, since we can have at most $\lfloor \frac{neededPairs}{2} \rfloor$ pairs. Therefore, the while loop executes at most $O(\log c)$ times, where c is the number of neighbors in BS's communication range, since it is the first and the greatest parameter that *CalculatePairs* takes. Since *CalculatePairs* runs in constant time, we should examine the running time of *MarkAndPair* method.

In Algorithm 7 (*MarkAndPair*), the operations out of the while loop run in constant time, therefore, we should calculate the running time of the while loop. The first loop at lines 6 to 12 runs in at most $O(c^2)$ time, since *T.length*, i.e., number of trees in the network can be at most c. The for loop at lines 19 to 25 also runs in $O(c^2)$. The critical part of this algorithm is the finding maximum matchings from the marked elements done in line 26. As explained before, the running time of this operation can be $O(c^4)$ at most when we use Edmond's Algorithm [27]. The running time can be improved if we use [28]'s solution as explained before. The maximum matching operation is repeated until the number of maximum matching is greater than or equal to number of pairs needed p. In every iteration, in the worst case, we can mark only two entries $M(i, j)$ and $M(j, i)$ at line 22, and calculate the maximum matching again. Therefore, in the worst case, $O(c^2)$ iterations are needed to reach to p maximum matchings. Consequently, the running time of the *MarkAndPair* Algorithm is calculated as $O(c^2 \times c^4) = O(c^6)$, when Edmond's Algorithm [27] is used for finding the maximum matching.

Since *MarkAndPair* Algorithm is called $O(\log c)$ times in the worst case in Algorithm 6, the overall running time of the *UniteTrees* Algorithm will be $O(c^6 \log c)$ when Edmond's Algorithm [27] is used. Since we form c trees initially and use the same algorithm with GreedyPMIT in NCCA-N and NCCA-D, and a similar one in BUCA-N and BUCA-D, the time complexity of the tree formation is $O(dcn^2)$. Consequently our final complexity will be $O(c^6 \log c + dcn^2)$. The complexity of our algorithms is worse than GreedyPMIT in the worst case, where $c > k$, however, the performance results of our algorithm are better than GreedyPMIT, which are explained in the next chapter.

Chapter 4

Performance Evaluation

In this chapter, we present our simulation environment and our simulation experiments we performed to evaluate our algorithms. For evaluation, we compare our algorithms (BUCA and NCCA) with GreedyPMIT [3], and show the improvements.

4.1 Simulation Environment and Scenarios

Our simulations are run on a Linux machine with 8 core 64-bit processor and 4 GB memory. Simulation is coded in Java, and run on 64-bit Java Runtime Environment (JRE). A remote server is used for experiments in order to run experiments in parallel and to save time. Although the server has 8 cores, mostly only 1 core is used for each experiment.

In the simulations, sink node (BS) is placed in the middle point, and the other nodes are placed around it. The nodes are placed so that every node has four nodes around it (except the nodes at the edges), which have exactly the same distance to the center node. These four nodes are placed at the left, right, top and bottom of the node. The distance between these four nodes and the center node is defined as 1 unit distance. The network structure is a grid, where nodes

are placed at the intersection points, and the BS is located at the middle point, and the network used in the simulations is a grid with a 100% density, which denotes that there is a node in every intersection point in the grid as seen from Figure 3.2. Although it has 100% density, the calculated interference and the tree formations can be changed for each run, because there are randomness in the algorithms, where there are tie breaks for the scenarios like each node having the same number of parents while sorting, or more than one tree has the same interference after adding a node, etc. Therefore, we have 100 repeated runs with different network formations for each scenario to achieve stable results.

In the experiments, we vary the number of nodes, communication range, interference range, and the number of channels available, and try to evaluate the performance of our algorithms. Although we set the communication range as 1, 1.5 and 2 unit distances, we only demonstrate the results for 1.5 and 2 units, since the results of our algorithm is mostly the same with GreedyPMIT when we set the communication range as 1 unit. For 1 unit case, since the communication range is low, the nodes do not have much option when selecting their parents, i.e., tree, and therefore, the formed trees in both algorithms are pretty much the same, which brings minor performance difference.

Another parameter is the interference range, and it is also defined as 1.5 times of the communication range in [3]. We also set it as 2 times of the communication range, however, we only show the results with 1.5 times. Although 2 times performs slightly better in some cases, we prefer 1.5 times, because it is used as 1.5 times in [3]. Since we are comparing with their algorithm, using their parameter value will allow a fair comparison.

The other important variable is the number of nodes in the network. We set the network as an $x \times x$ network, and choose x to be an odd number, which will make the BS located exactly at the middle of the network. The variable x is set to odd numbers between 11 and 33, inclusively, therefore, the number of nodes are set to 121, 169, 225, 289, 361, 441, 529, 625, 729, 841, 961 and 1089.

Final variable is the available number of channels, which is also equal to the number of trees in the network. Although there are 16 channels in 2.4 GHz in ZigBee, [3] demonstrates that adjacent channels cause interference, and decrease the overall throughput. Therefore, it will be better to use non-adjacent channels, i.e., maximum of 8 channels can be used. Consequently, we choose the number of channels between 2 to 8, inclusively, in our simulations. As expected, the interference in the network with less channels will be more, and interference will decrease when the number of channels increases in both our algorithms and GreedyPMIT. In the performance evaluation, we both present the actual interference values and the performance improvement compared to GreedyPMIT. The performance improvement is shown with interference decrease in percentage in Figures 4.1 to 4.8 and actual interference values can be seen in Figures 4.9 to 4.12. Although our algorithms perform better than GreedyPMIT in general, there can be some points that GreedyPMIT performs better than our algorithms, and these results are demonstrated as negative interference decrease in Figures 4.1 to 4.8.

As stated in the previous chapters, GreedyPMIT-N is the algorithm proposed in [3], also known as GreedyPMIT, which is our base algorithm, and we try to achieve better results than that. The other algorithms with suffix -N, i.e., BUCA-N and NCCA-N, are proposed by us, and perform bottom-up and top-down tree formation, respectively, and at the end they both unite the initial trees (if needed) to form k trees, where k is the available number of channels. The algorithms with suffix -D are also proposed by us, and are the same algorithm with the same prefix (ex: BUCA-N and BUCA-D), the only difference is the interference metric they use while trying to improve the performance. Algorithms with suffix -N use the number of same tree nodes in its interference range, however, algorithms with suffix -D use the distance between nodes as interference metric, as explained before. In the following section, we present the simulation results of our algorithms by comparing BUCA-N and NCCA-N with GreedyPMIT-N, and BUCA-D and NCCA-D with GreedyPMIT-D.

4.2 Simulation Results

In this section, we present and discuss simulation results. We examine our simulation results depending on the value of the communication range variable, in Sections 4.2.1 and 4.2.2. We also compare our six algorithms with respect to the distance-based interference metric, which we think is a more accurate metric than the node-count based metric, in Section 4.2.3.

4.2.1 Communication Range = 1.5 units

The first part of the simulation results are for the case where the communication range is set to be 1.5 units. As explained earlier, the interference range is fixed and set as 1.5 times of the communication range. Hence, it is set to be 2.25 units. When we set the communication range as 1.5 units, a node can have at most 8 nodes in its one-hop communication range and 20 nodes in its interference range. This can be observed from Figure 3.2, where inner dashed circle denotes the communication disk, and outer dashed circle denotes the interference disk of a node. With 1.5 unit communication range, the number of neighbors of the BS, i.e., the value of c, will also be at most 8. As explained in the previous section, we examine our four algorithms against GreedyPMIT-N and GreedyPMIT-D while changing the number of available channels from 2 to 8, inclusively.

4.2.1.1 Comparisons with GreedyPMIT-N

In this section, we present the comparison of GreedyPMIT-N with our algorithm BUCA-N, and GreedyPMIT-N with our algorithm NCCA-N. Although both of our algorithms have different approaches for tree formation, they exhibit similar performance results in most cases.

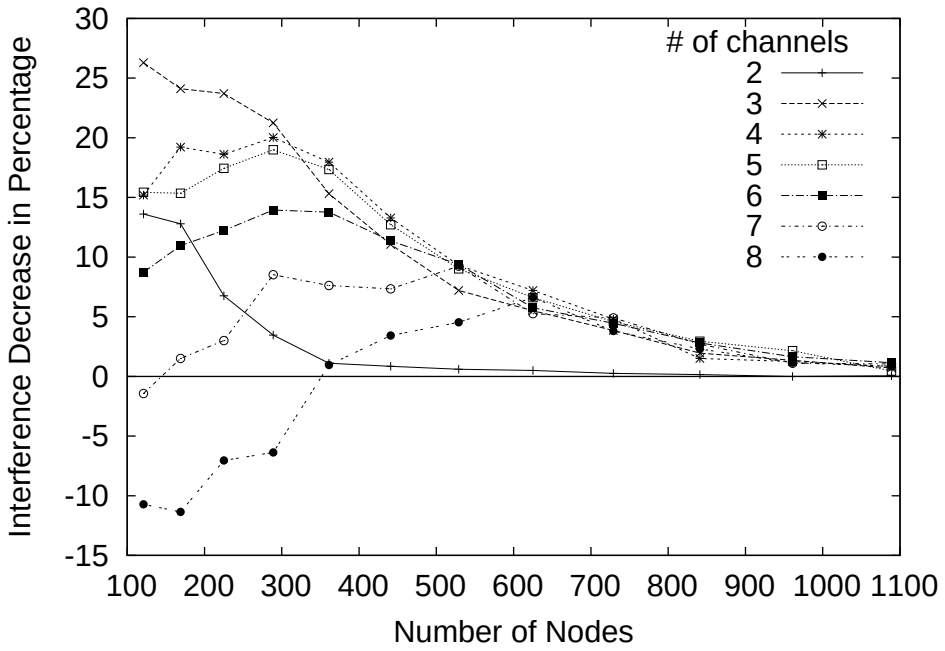

Figure 4.1: Comparison of GreedyPMIT-N and BUCA-N when communication range = 1.5 units. The y-axis is interference decrease, i.e., performance improvement, of our algorithms against GreedyPMIT.

Figure 4.1 demonstrates the comparison between GreedyPMIT-N and BUCA-N, when communication range is 1.5 units. The performance increase in the figure can be observed from y-axis, which is interference decrease in percentage when compared to GreedyPMIT-N. This figure shows that, except for the case where channel count is 2, when the number of channels increases, the performance improvement for smaller networks decreases, since the number of union operations decreases. For example, number of union operations will be 5 for 3 channels, only 1 for 7 channels, and also there will not be any union operation when the number of channels is 8, when communication range = 1.5 units. It shows that the union operation plays a big role in the performance of our algorithm. When the number of nodes increases, the performance depending on channel count will be nearly equal, and diverges to zero, i.e., shows a similar performance with GreedyPMIT. The reason of the performance decrease of the case, where channel count is 2, can be due to the high interference. With 2 available channels, we will have only 2 trees, therefore, the interference may not be improved much.

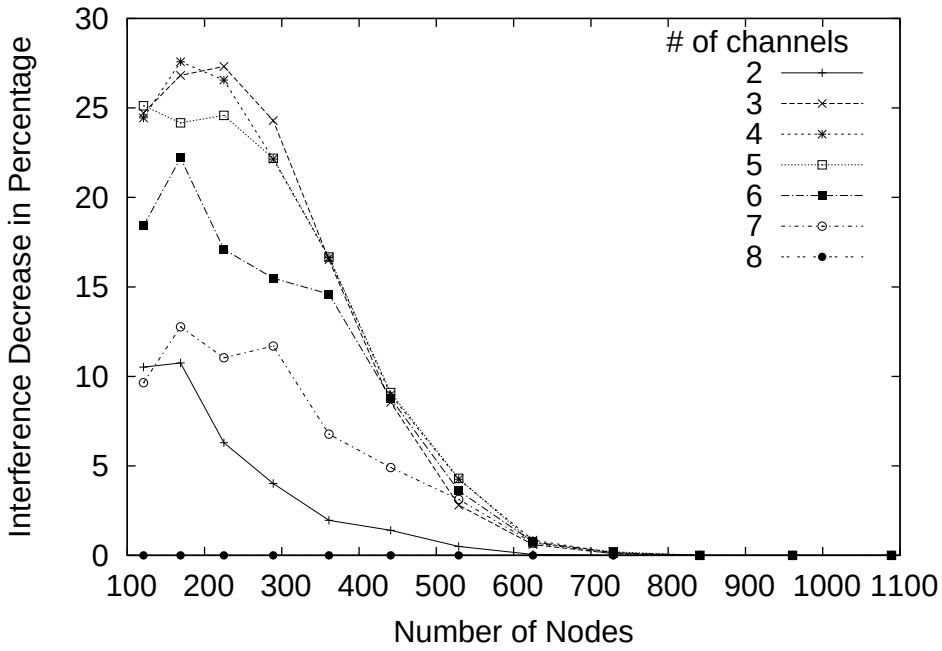

Figure 4.2: Comparison of GreedyPMIT-N and NCCA-N when communication range = 1.5 units.

In Figure 4.2, we present the performance improvement of Algorithm NCCA-N against GreedyPMIT-N when communication range is 1.5 units. First of all, we notice that 8 channels case do not have any improvement, it performs the same with GreedyPMIT-N. Because for 1.5 units communication range, BS will have 8 neighbors, i.e., $c = 8$, and our algorithm forms c trees in the same way with GreedyPMIT-N. Then our algorithm unites the trees to decrease the number of trees to k trees. In the 8 channel scenario, both c and k are equal to 8, consequently there will not be any union operation, and the interference value of the network will be the same with GreedyPMIT.

As stated before, using 2 channels in BUCA-N do not show much improvement, although it has more union operations than the others, i.e., 6 operations. We think that the reason is the unavoidable high interference due to having only two trees, despite the high number of union operations. This property is also seen is Figure 4.2.

The other similarity between BUCA-N and NCCA-N is that, networks with less channels perform better when we compare them with 6, 7 and 8 channels. Since 6, 7 and 8 channels have 2, 1 and 0 union operations and 3, 4, 5 channels have 5, 4 and 3 union operations, respectively. A difference between BUCA-N and NCCA-N is that, except the 2 channels case, the performance increase in NCCA-N is higher in smaller sized networks, however, the performance decreases faster when the network size increases, and reaches 0 when the node size is equal to 729, however, in BUCA-N, the performance only diverges to 0 in any node size. Also in these simulations the performance improvement can be as much as 30% when compared with GreedyPMIT.

4.2.1.2 Comparisons with GreedyPMIT-D

The comparisons of GreedyPMIT-D with BUCA-D, and GreedyPMIT-D with NCCA-D is presented in this section. First, the comparison between GreedyPMIT-D and BUCA-D will be discussed.

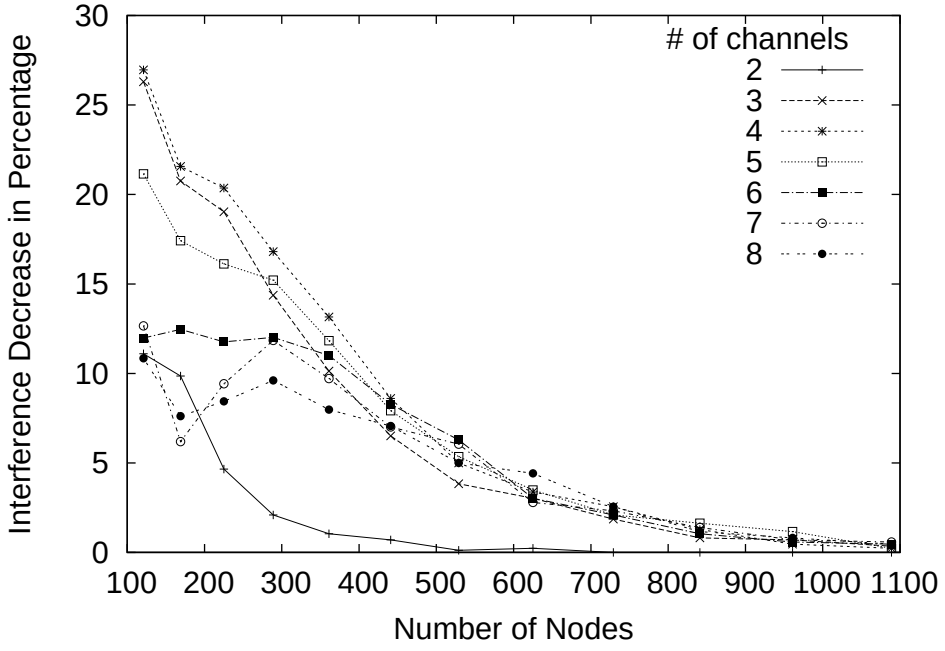

Figure 4.3: Comparison of GreedyPMIT-D and BUCA-D when communication range = 1.5 units.

In GreedyPMIT-D, BUCA-D and NCCA-D the only difference is the interference metric when we compare with algorithms GreedyPMIT-N, BUCA-N and NCCA-N, respectively. In algorithms with suffix -D the interference metric is not the number of same tree nodes in the interference disk, it is the sum of the distances to the other same tree nodes in a node's interference disk.

Figure 4.3 demonstrates the performance improvement of BUCA-D against GreedyPMIT-D when communication range is 1.5 units. In this figure, we can observe a similar performance for 2 channels case with previous two figures. And also the performance difference between 3 to 5 and 6 to 8 channels cases can be seen from this figure, too.

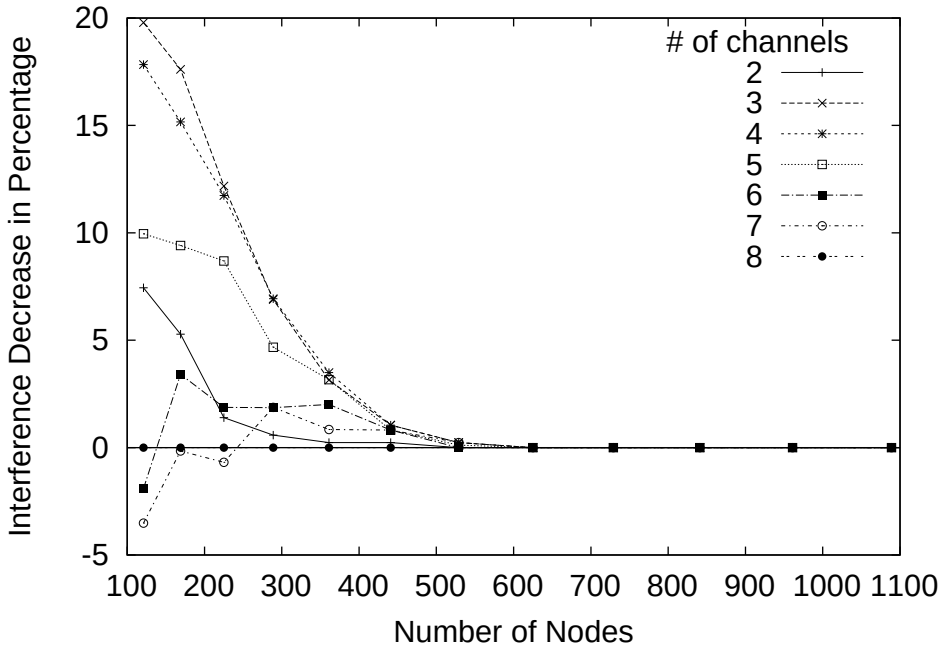

Figure 4.4: Comparison of GreedyPMIT-D and NCCA-D when communication range = 1.5 units.

Figure 4.4 demonstrates the performance improvement of NCCA-D against GreedyPMIT-D when communication range is 1.5 units. NCCA-D is another version of NCCA-N, where the only difference is the interference metric. Therefore, it is natural to expect a similar result with NCCA-N. As seen from Figures 4.2 and 4.4, they are very similar. First similarity is the performance of 8 channels case, due to the tree formation procedure. However, NCCA-D performs a little

worse than NCCA-N. Moreover, for 6 and 7 channels cases, NCCA-D performs worse than GreedyPMIT-D for smaller networks. And also the performance difference between GreedyPMIT-D and NCCA-D reaches to 0 when network size is greater or equal to 625.

4.2.2 Communication Range = 2 units

For communication range = 2 units, there will be 12 other nodes in a node's vicinity, that can be reached via 1-hop, therefore, number of neighbors of BS is 12. And also the number of nodes in a node's interference range is 28 with interference range = $2 \times 1.5 = 3$ units. On the other hand, the number of nodes in communication range and interference range is 8 and 20, respectively, for communication range = 1.5 units. Consequently, higher communication range causes increase in the interference due to the increase in the number of nodes in the interference range. In the following set of simulation results, we observe the performance of our algorithms when communication range is 2 units.

4.2.2.1 Comparisons with GreedyPMIT-N

We present the comparisons of BUCA-N and GreedyPMIT-N, and NCCA-N and GreedyPMIT-N in this section.

Figure 4.5 compares the performance between GreedyPMIT-N and BUCA-N when communication range is 2 units. In this figure, we encounter a very different result than Figure 4.1, which represents the comparison between the same algorithms when communication range = 1.5 units. In Figure 4.1, the performance decreases when the number of nodes increases, however, in this scenario, where communication range is 2 units, we observe the opposite result, where the increase in the number of nodes results in increase in the performance. The number of nodes in one node's interference range is more than the previous scenario, and this number plays an important role in the interference, and it seems that

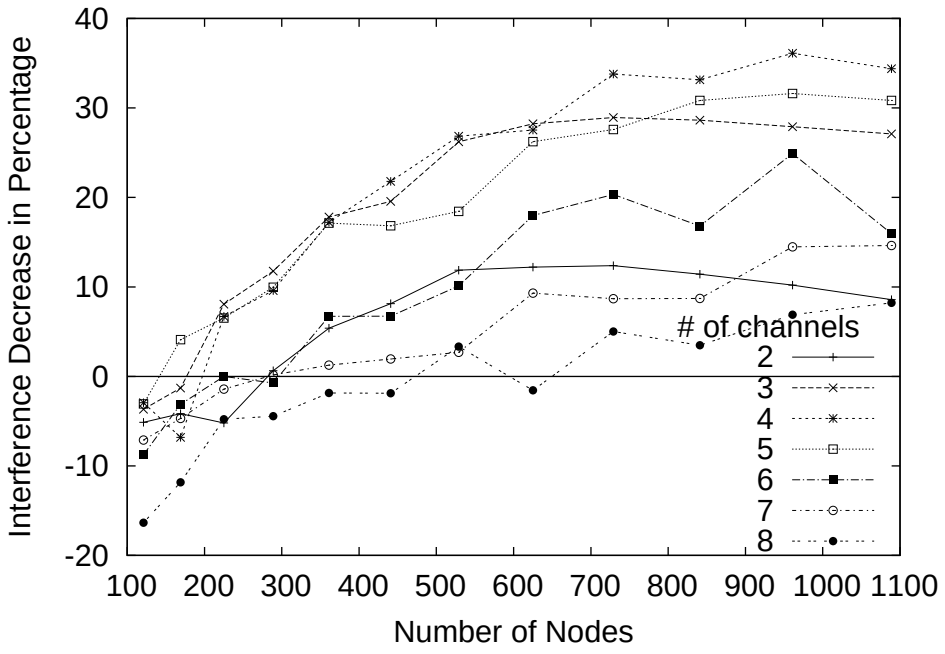

Figure 4.5: Comparison of GreedyPMIT-N and BUCA-N when communication range = 2 units.

this is an overhead for smaller networks. However, when the number of nodes increases, the effect of that overhead also decreases and BUCA-N starts to perform better than GreedyPMIT-N. As seen from the figure, except the 8 channels case, our algorithm performs better than GreedyPMIT, when the number of nodes are greater than or equal to 361, and 8 channels case starts to perform better in larger networks. If we omit the 2 channels case again, the higher performance of less number of channels draw the attention. Although larger networks do not have any improvement in the performance for 1.5 units communication range, our algorithms achieve remarkable improvements with 2 units communication range by achieving 36% performance increase at most.

Figure 4.6 compares the performance between NCCA-N and GreedyPMIT-N when communication range is 2 units. In Figure 4.6, in general, we observe better results than Figure 4.5 in both smaller networks and larger networks. First, there is not any negative result except the result for 2 channels case with 121 nodes. In the rest, our algorithm performs better than GreedyPMIT, and achieves as much as 40% performance increase. An interesting result from this figure is that

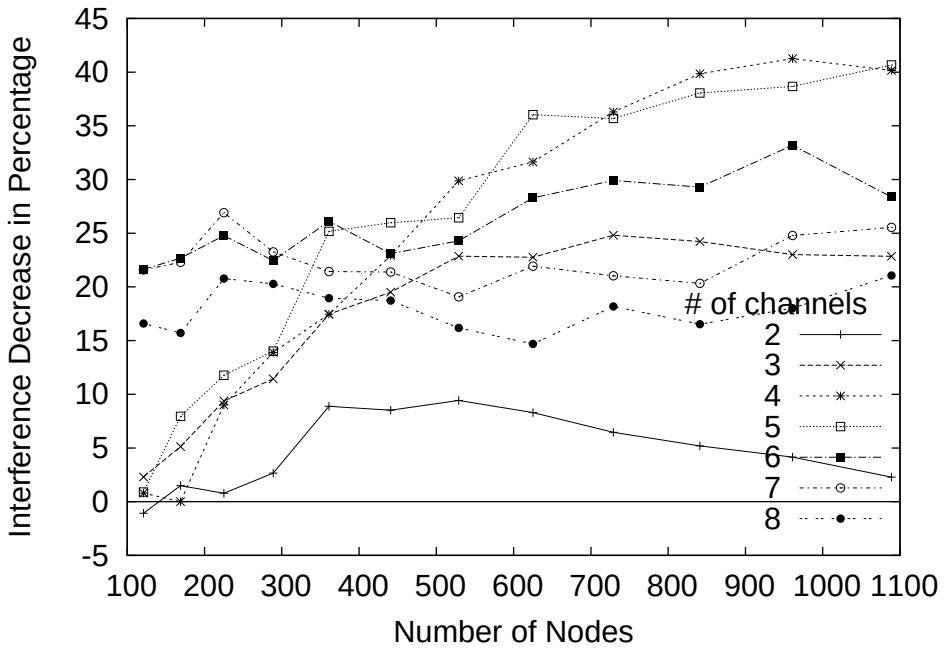

Figure 4.6: Comparison of GreedyPMIT-N and NCCA-N when communication range = 2 units.

3 channels case performs worse than 4, 5 and 6 channels cases, especially when the network size increases, although it has more union operations. The reason should be similar with 2 channels cases' performance, which is explained before. Another important observation is that 6, 7 and 8 channels cases perform in a more stable way, however, 3, 4 and 5 channels cases start with lower performance when the number of nodes are smaller, and show a huge improvement when the number of nodes increases, especially in 4 and 5 channels cases, by achieving 40% performance increase.

4.2.2.2 Comparisons with GreedyPMIT-D

Now, we present the comparisons of BUCA-D with GreedyPMIT-D and NCCA-D with GreedyPMIT-D, which use the distance as interference metric.

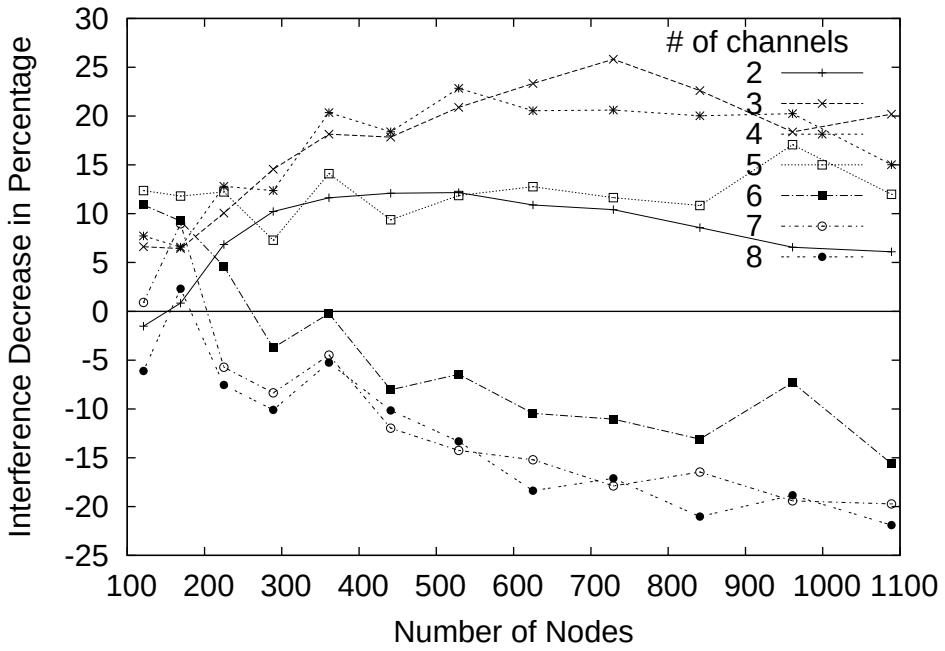

Figure 4.7: Comparison of GreedyPMIT-D and BUCA-D when communication range = 2 units.

Figure 4.7 compares the performance between BUCA-D and GreedyPMIT-D when communication range is 2 units. The results in Figure 4.7 are a little disappointing especially for the larger networks with 6 to 8 channels cases. Those results are negative valued, and can reach 22% performance decrease. On the other hand, it performs better in 2 to 5 channels cases, and 2 channels case performs the worst in those. Except the 6 to 8 channels cases, this figure resembles Figure 4.5, which is another version of the same algorithm with different interference metrics.

Figure 4.8 compares the performance of GreedyPMIT-D and NCCA-D when communication range is 2 unit, which is similar to Figure 4.6. We can state that NCCA-D performs mostly better than GreedyPMIT by achieving positive interference decrease. Similar with the comparison of GreedyPMIT-N and NCCA-N with communication range = 2, 2 and 6 to 8 channels cases show more stable results, when compared to 3 to 5 channels cases, where their performance increases when the network size increases. And again 2 channels do not show much performance improvement due to the high interference caused by two physically

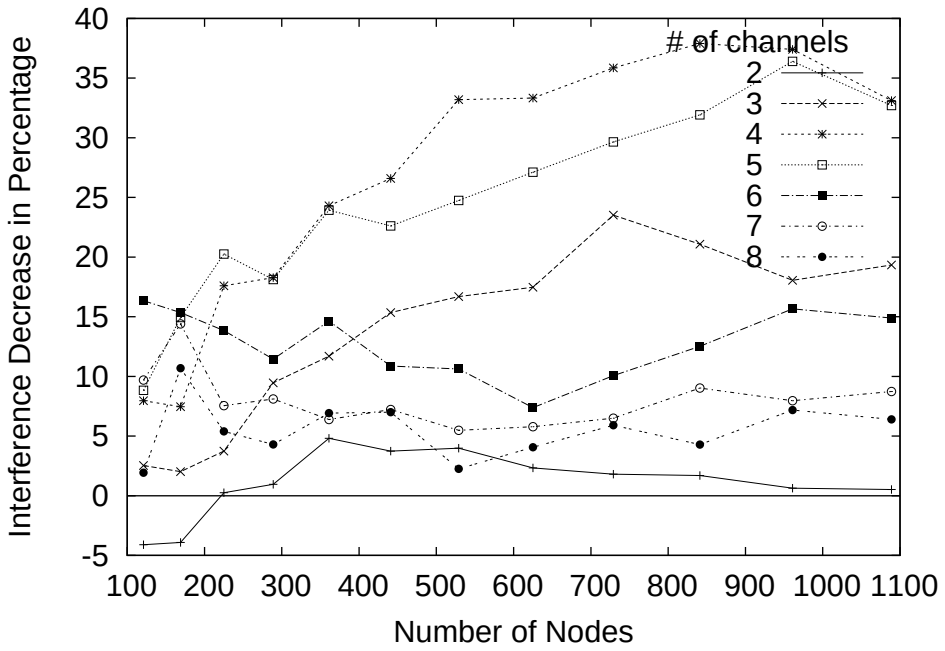

Figure 4.8: Comparison of GreedyPMIT-D and NCCA-D when communication range = 2 units.

close trees.

4.2.3 Comparison of All Algorithms when Distance-Based Interference Metric is Used for Calculating the Final Interference

Since we take GreedyPMIT as our base algorithm, algorithms with a suffix -N uses number of nodes in the interference disk, which is the default interference metric of GreedyPMIT. Although using this metric is plausible and easy for implementation, additionally the distance between the nodes in the interference disk is also important in real world, that is why we propose distance-based interference metric. Therefore, we compare the interferences of the final trees formed from all six algorithms, where four of them are new algorithms proposed by us, by using distance-based interference metric, regardless of their tree formation metrics. Although the algorithms with suffix -N use number of nodes as their interference

metric, their final performance will be compared by using distance-based interference metric, to simulate the real world, where not only the number of nodes, but also the distance between the nodes are significant. Unlike the previous figures in this chapter, we show the actual interference values in the network, not the performance increase, where the performance will be higher, if the interference value is lower.

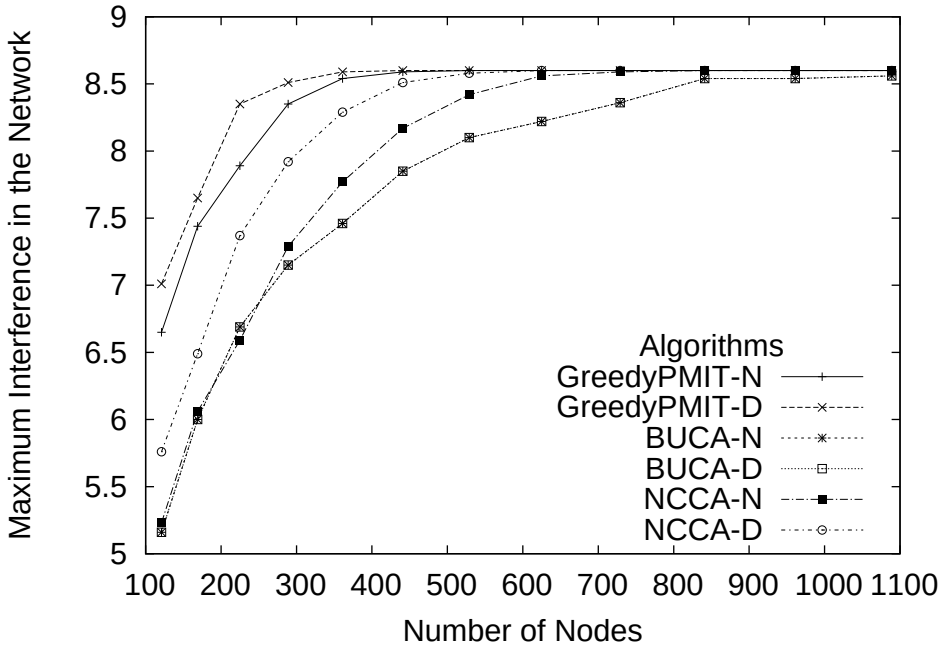

Figure 4.9: Comparison of all algorithms, when the final interference is calculated using distance-based interference metric, where $k = 4$, and communication range = 1.5 units.

Figure 4.9 shows the final maximum interferences of the networks with different number of nodes, when the final interference is calculated using distance-based interference metric, regardless of the metric used in tree formations. In this figure, communication range = 1.5 units and available number of channels, i.e., k = 4. The results are very similar for channel counts between 2 and 8, inclusively, when communication range = 1.5 units. Therefore, we show only the results of the 4 channels case, where the difference between each algorithm can be seen better.

Since we show that our proposed algorithms (BUCA and NCCA) mostly perform better than GreedyPMIT, the interference values of GreedyPMIT-N is higher than BUCA-N and NCCA-N, and interference values of GreedyPMIT-D is higher than BUCA-D and NCCA-D, as shown in Figure 4.9, where higher interference indicates lower performance. As seen from Figures 4.1 to 4.4, there is not any performance improvement for larger networks when communication range = 1.5 units, therefore, the interference values for larger networks are close to each other in this figure.

In Figure 4.9, for the algorithms except BUCA-N and BUCA-D, the networks with greater than or equal to 841 nodes reach to 8.6 interference value, which is the maximum interference that can be achieved, when communication range is set as 1.5 units, when all the nodes in the interference disk is the same-tree nodes for at least one node in the network. It is the worst-case tree formation, and cannot be avoided for larger networks apparently.

An interesting result for Figure 4.9 is that, although NCCA-N and GreedyPMIT-N do not form the trees according to distance-based interference metric, their interference values are less than their distance-based interference metric versions NCCA-D and GreedyPMIT-D, respectively, in smaller networks. Although this is not expected, this result can be explained with the greedy property of the algorithms, where the best parent of a node for a level is chosen when trees are formed, and cannot be changed in the next levels.

We demonstrate the results of the simulation results, where communication range = 2 units into three parts, since the results in that case are not similar in different k values, unlike previous case (communication range = 1.5 units). This case is examined with three channel cases, where $k = 3$, 5 and 7, in the increasing order to observe the effect of the number of channels to the performances of the algorithms better. We start with the results of 3 channels case.

Figure 4.10 shows the final maximum interferences of the networks when the final interference is calculated using distance-based interference metric, when k = 3 and communication range = 2 units. GreedyPMIT-N and GreedyPMIT-D again have the highest interference values. Contrary to Figure 4.9, in this figure,

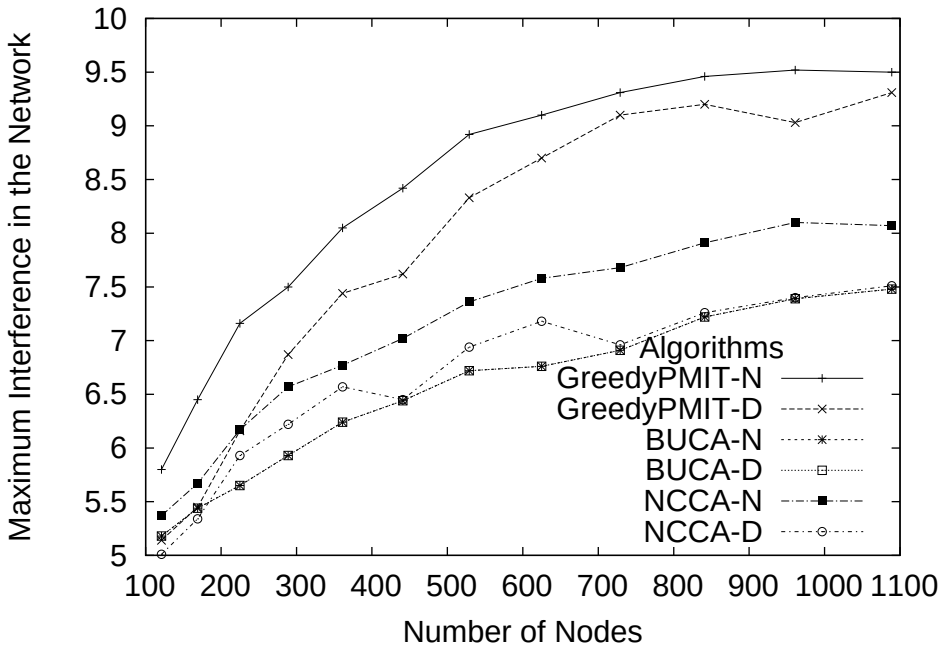

Figure 4.10: Comparison of all algorithms, when the final interference is calculated using distance-based interference metric, where $k = 3$, and communication range = 2 units.

algorithms with distance-based interference metric, suffix -D, show better or equal performances when compared with the same named algorithms with suffix -N, as expected. Also, only GreedyPMIT algorithms can reach interference values close to the maximum interference, which is approximately 9.55, when communication range = 2 units. Additionally, the performance increase can be observed better in this figure, where there are more interference difference between GreedyPMIT algorithms and our algorithms.

Figure 4.11 shows the final maximum interferences of the networks when the final interference is calculated using distance-based interference metric, when k = 5 and communication range = 2 units. We can observe the actual interference differences between 3 and 5 channels cases, by examining Figures 4.10 and 4.11, respectively. As expected, interference values in 5 channels case is lower than 3 channels case, since the network is divided into more trees in 5 channels case, consequently the number of nodes per each tree decreases, and this causes lower interference than 3 channels case. Since we have more trees in this figure, the

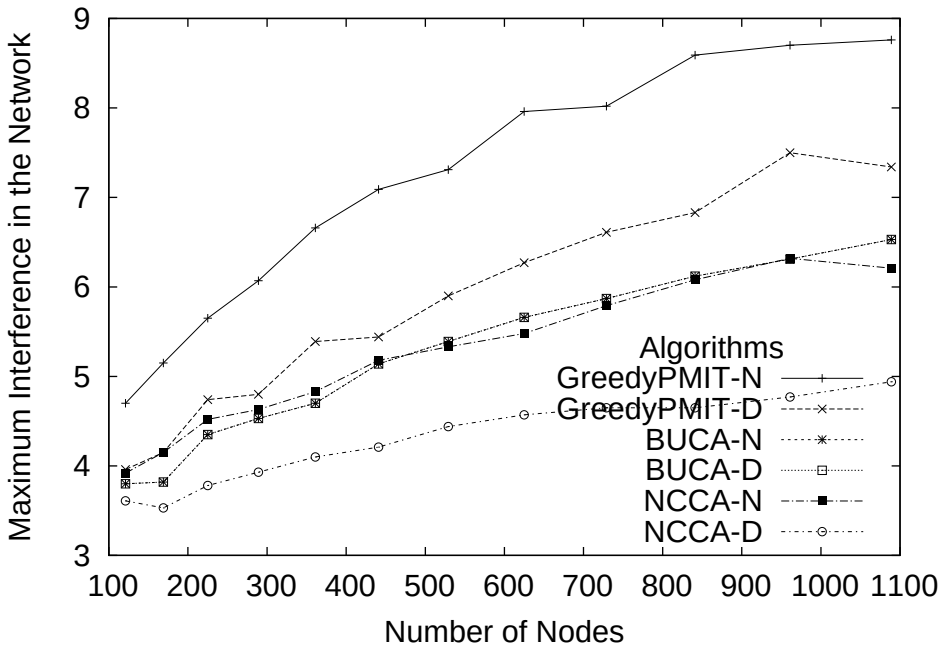

Figure 4.11: Comparison of all algorithms, when the final interference is calculated using distance-based interference metric, where $k = 5$, and communication range = 2 units.

maximum interference values in the network do not reach to the exact maximum interference value. We can also observe that the performance of BUCA algorithms are decreased when compared to NCCA-D algorithm, when the number of available channels increases.

Figure 4.12 shows the final maximum interferences of the networks when the final interference is calculated using distance-based interference metric, when k = 7 and communication range = 2 units. The decrease of interference values according to the increase in the available number of channels can be clearly observed in Figures 4.10 to 4.12. Also performance decrease of BUCA algorithms and performance increase of GreedyPMIT-D when number of available channels increases draw attention.

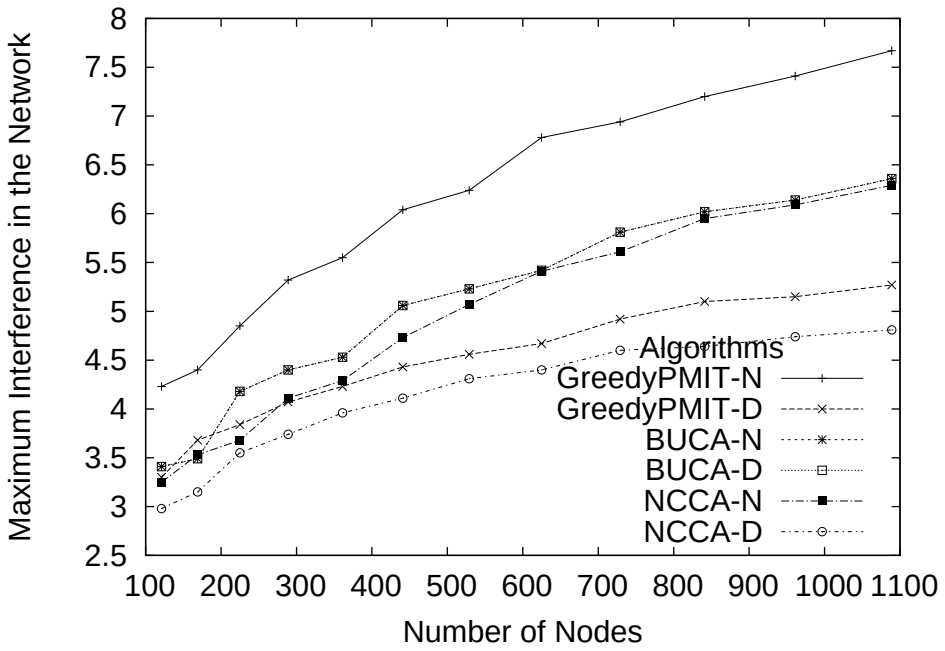

Figure 4.12: Comparison of all algorithms, when the final interference is calculated using distance-based interference metric, where $k = 7$, and communication range = 2 units.

4.2.4 Summary

As a conclusion for the simulation results with 1.5 units communication range, naturally less number of channels perform better than the others due to the more number of union operations. There is an exception that channel 2 do not perform much better, one reason can be due to the unavoidable high interference, which is the result of using only two trees. Another point is that, when the number of nodes in the network increases, the performance decreases, and can reach to 0. Although there are a few results that our algorithms perform worse than GreedyPMIT, in the most of the results, our algorithm performs better than GreedyPMIT, and reaches maximum of 27.5% performance increase.

Maximum interference decrease, when the communication range is 2 units, 40%, is higher than the maximum interference decrease when the communication range is 1.5 units, 27.5%. The higher interference decreases are obtained from the larger networks when communication range is 2 units, where the overhead of the

high interference range, i.e., higher number of nodes in the interference range, is eliminated. In general, 3 to 5 channels cases perform better than 6 to 8 channels cases, due to the increase in the number of union operations, similar with the previous set of experiments. Also, in one experiment, our algorithm performs worse than GreedyPMIT with distance-based interference metric when we have 6 to 8 channels.

In general, our newly proposed algorithms with each interference metric show better performance than GreedyPMIT with our proposed union operation, which forms more trees at the beginning, and unites the trees in order to decrease the number to k trees. Another inference from these simulations is that, using communication range as 2, rather than 1.5 units brings a great increase in the performance, especially for larger networks. Oppositely, using communication range as 1.5 units will be better for smaller networks as explained before.

The two types of algorithms proposed show similar results, however, the poor performance of BUCA-N with 6 to 8 channels cases when communication range = 2 disappoints and motivates us to propose our other algorithm NCCA-N, which performs better than BUCA-N in that case.

For the second type of simulations described in Section 4.2.3, where distance-based interference metric is used for evaluating all of the six algorithms, we can clearly observe the decrease in the actual interference values, when available number of channels increases. The main reason of this is the decrease in the number of nodes per trees. As expected, the algorithms which use number of nodes, especially GreedyPMIT-N, cannot perform well in this case where real world effects are considered. We also observe from these simulations that, the performance of the algorithms are also affected from the number of channels available, where BUCA algorithms performs much better than NCCA for larger k values.

Chapter 5

Conclusion and Future Work

In this thesis, we propose channel assignment algorithms that can be used to decrease the interference caused by co-channels and adjacent channels in multi-channel wireless sensor networks. In this way we are trying to increase the network throughput and reduce number of collisions and packet corruptions. To do this, we try to improve the performance of an existing channel assignment algorithm, GreedyPMIT [3], which divides the sensor network rooted at the base station into k trees, where k is the number of available channels, and assigns unique non-overlapping (orthogonal) channels to these trees. In this way, the inter-tree interference is eliminated and the only interference that remains is intra-tree interference. Intra-tree interference to a node is caused by other nodes that are close to that node and in the same tree.

The interference of the network is calculated in [3] as the maximum interference received by a non-leaf node. The leaf nodes are not counted, since they use a receiver centric interference definition. The interference received by a node is defined as the number of same-tree nodes in its interference range. When forming the trees, this interference value is used. The channel, i.e., tree assignment starts from the nodes which have one-hop distance from BS, i.e., level-one nodes. Nodes choose their parent formed from a BFS Fat Tree Algorithm, which has the least interference after adding that node. This procedure repeats until the lowest level is reached. At the end, the network is divided into k trees with a minimal

intra-tree interference.

First we implement GreedyPMIT in Java and prepare a graphical representation of the network in order to examine how GreedyPMIT can work better. We identify the weak points of the algorithm and devise ideas to improve it further. In GreedyPMIT, due to the greedy property of the algorithm, a node cannot change the tree it has been assigned in previous levels. Consequently, GreedyPMIT can geographically cluster the nodes of a tree; as a result, lots of nodes in the same tree can gather in the same region. This causes high interference, since it increases the number of same-tree nodes in a vicinity.

To increase the physical separation of nodes belonging to the same tree, we propose another interference metric, which uses the distance between the nodes. Because, we know that the distance between nodes also affect the interference. Therefore, we can have two versions of every algorithms we proposed, i.e., with the default interference metric and the distance-based interference metric.

Then we propose a new channel assignment algorithm, that is similar to GreedyPMIT in some aspects, but which has the tree formation performed bottom up, contrary to GreedyPMIT, which is performing top-down tree formation. We call this algorithm as Bottom Up Channel Assignment (BUCA). Another major difference between BUCA and GreedyPMIT is that BUCA is not forming k trees initially, but is forming c trees, where c is the number of nodes in BS's communication range. In this way, the hot-spot problem around the base station is solved (all its children are using a different channel). Then we propose a way to unite these c trees to decrease the number to k, if needed. We propose, two versions of BUCA, one using node-count to compute interference (BUCA-N), the other using distance to nodes as well (BUCA-D).

We also propose another algorithm for assigning channels. This second algorithm assigns channels top-down, as in GreedyPMIT, but creates initially c number of trees as in BUCA. Hence it is similar to both GreedyPMIT and BUCA. We call this algorithm as Neighbor Count based Channel Assignment (NCCA). It also has two versions: NCCA-N (uses node-count based interference metric) and NCCA-D (uses distance-based interference metric).

To evaluate the performance of our four algorithms (BUCA-N, BUCA-D, NCCA-N, NCCA-D), we run extensive simulations. In our simulations, we change the available channels between 2 and 8. We vary the communication range between 1.5 and 2. We test the algorithms for various network sizes. The results show that our algorithms perform mostly better than GreedyPMIT. Although there are some results with poor performance, dominantly our algorithm performs better than GreedyPMIT and can achieve as much as 40% performance increase. We also compare all six algorithms (GreedyPMIT-N, GreedyPMIT-D, and our four algorithms) using a distance-based interference metric for comparison. All results show that our algorithms perform better in the majority of the cases.

5.1 Future Work

In this thesis, we work in WSNs with a single sink (BS), where all the nodes send their data packets to it. Our algorithms can be modified to handle multi-sink cases as well. Each sink can run a similar algorithm to the one-sink case. But then, trees rooted at different base stations can have the same channel. To prevent this, several additions can be made to our algorithms like decreasing the number of available channels when moving away from a BS to decrease the interference in regions where trees from different BSs meet each other.

Bibliography

[1] "TelosB datasheet." `http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf`. Last access: 4 September 2012.

[2] "Graphical representation of Wi-Fi channels in the 2.4 GHz band." `http://en.wikipedia.org/wiki/File:2.4_GHz_Wi-Fi_channels_(802.11b,g_WLAN).svg`. Last access: 4 September 2012.

[3] Y. Wu, J. A. Stankovic, T. He, and S. Lin, "Realistic and efficient multi-channel communications in wireless sensor networks," in *INFOCOM 2008. The 27th Conference on Computer Communications, IEEE*, pp. 1193 –1201, April 2008.

[4] O. Durmaz Incel, *Multi-channel wireless sensor networks: protocols, design and evaluation*. PhD thesis, Enschede, March 2009.

[5] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, no. 4, pp. 393 – 422, 2002.

[6] J. Yick, B. Mukherjee, and D. Ghosal, "Wireless sensor network survey," *Computer Networks*, vol. 52, no. 12, pp. 2292 – 2330, 2008.

[7] E. I. Oyman and C. Ersoy, "Multiple sink network design problem in large scale wireless sensor networks," in *IEEE International Conference on Communications*, vol. 6, pp. 3663 – 3667, June 2004.

[8] "ZigBee specification." http://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/kjb79_ajm232/pmeter/ZigBee%20Specification.pdf, January 2008. Last access: 4 September 2012.

[9] "IEEE 802.15.4-2003 standard." http://standards.ieee.org/getieee802/download/802.15.4-2003.pdf, October 2003. Last access: 4 September 2012.

[10] "CC2420 2.4 GHz IEEE 802.15.4 / ZigBee-ready RF transceiver." http://www.ti.com/lit/ds/symlink/cc2420.pdf, 2007. Last access: 4 September 2012.

[11] "IEEE 802.11-2007 standard." http://standards.ieee.org/getieee802/download/802.11-2007.pdf, June 2007. Last access: 4 September 2012.

[12] "nRF905 single chip 433/868/915 MHz transceiver." http://www.nordicsemi.com/eng/content/download/2452/29528/file/Product_Specification_nRF905_v1.5.pdf, April 2008. Last access: 4 September 2012.

[13] "CC1000 single chip very low power RF transceiver." http://www.ti.com/lit/ds/symlink/cc1000.pdf, 2007. Last access: 4 September 2012.

[14] "TR1001 868.35 MHz hybrid transceiver." http://www.rfm.com/products/data/tr1001.pdf, 2008. Last access: 4 September 2012.

[15] "TDA5250 D2 ASK/FSK 868 MHz wireless transceiver." http://www.infineon.com/dgdl/TDA5250_DS_V1.7.pdf?folderId=db3a30431689f4420116a096e1db033e&fileId=db3a3043191a246301192e72a7312c03, February 2007. Last access: 4 September 2012.

[16] "Cisco systems HWIC-AP data sheet." http://www.cisco.com/asiapac/campaigns/isr/files/datasheet/hwic_ap_isr_ds_v1.pdf, 2005. Last access: 4 September 2012.

[17] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of the 10th Annual International Conference*

on Mobile Computing and Networking, MobiCom'04, (New York, NY, USA), pp. 114–128, ACM, 2004.

[18] K. N. Ramachandran, E. M. Belding, K. C. Almeroth, and M. M. Buddhikot, "Interference-aware channel assignment in multi-radio wireless mesh networks," in *INFOCOM 2006. Proceedings of the 25th IEEE International Conference on Computer Communications*, pp. 1 –12, April 2006.

[19] Y. Jeong, J. Kim, and S.-J. Han, "Interference mitigation in wireless sensor networks using dual heterogeneous radios," *Wireless Networks*, vol. 17, pp. 1699–1713, October 2011.

[20] A. Dhananjay, H. Zhang, J. Li, and L. Subramanian, "Practical, distributed channel assignment and routing in dual-radio mesh networks," in *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, (New York, NY, USA), pp. 99–110, ACM, 2009.

[21] A. Raniwala and T. Chiueh, "Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network," in *INFOCOM 2005. Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*, vol. 3, pp. 2223 – 2234 vol. 3, March 2005.

[22] G. Zhou, C. Huang, T. Yan, T. He, J. A. Stankovic, and T. F. Abdelzaher, "MMSN: Multi-frequency media access control for wireless sensor networks," in *INFOCOM 2006. Proceedings of the 25th IEEE International Conference on Computer Communications*, pp. 1 –13, April 2006.

[23] J. Zhang, G. Zhou, C. Huang, S. H. Son, and J. A. Stankovic, "TMMAC: An energy efficient multi-channel MAC protocol for ad hoc networks," in *ICC '07. IEEE International Conference on Communications*, pp. 3554 –3561, June 2007.

[24] O. Durmaz Incel, "A survey on multi-channel communication in wireless sensor networks," *Computer Networks*, vol. 55, no. 13, pp. 3081 – 3099, 2011.

[25] J. So and N. H. Vaidya, "Multi-channel MAC for ad hoc networks: handling multi-channel hidden terminals using a single transceiver," in *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc'04, (New York, NY, USA), pp. 222–233, ACM, 2004.

[26] A. Mishra, E. Rozner, S. Banerjee, and W. Arbaugh, "Exploiting partially overlapping channels in wireless networks: turning a peril into an advantage," in *Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, IMC '05, (Berkeley, CA, USA), pp. 29–29, USENIX Association, 2005.

[27] J. Edmonds, "Paths, trees, and flowers," in *Classic Papers in Combinatorics* (I. Gessel and G.-C. Rota, eds.), Modern Birkhuser Classics, pp. 361–379, Birkhuser Boston, 1987.

[28] S. Micali and V. V. Vazirani, "An $o(\sqrt{|v|} \cdot |e|)$ algorithm for finding maximum matching in general graphs," in *21st Annual Symposium on Foundations of Computer Science*, pp. 17 –27, October 1980.