TECHNICAL UNIVERSITY OF CRETE
ELECTRONIC AND COMPUTER ENGINEERING DEPARTMENT
TELECOMMUNICATIONS DIVISION

Distributed Channel Allocation Algorithms for Wireless Sensor Networks

by

Efthymios Vlachos

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DIPLOMA DEGREE OF

ELECTRONIC AND COMPUTER ENGINEERING

September 2012

THESIS COMMITTEE

Thesis Supervisor:	Assistant Professor Aggelos Bletsas
Committee Member:	Professor Athanasios P. Liavas
Committee Member:	Professor Michael Paterakis

Abstract

As wireless sensor networks (WSNs) become larger and denser, the use of a single frequency among its nodes may cause severe interference problems and thus, dramatic packet losses, low packet delivery ratios and extensive delays. Since the half-duplex radio tranceiver of current WSN hardware can operate on multiple frequencies, the idea of effectively using them in order to improve the network's performance sounds quite attractive.

In this thesis, two well-known distributed channel allocation protocols found in the literature are examined under a signal to interference-plus-noise ratio (SINR) model. The first is a link-based fair channel allocation scheme that minimizes the maximum interference suffered by any link of the network, called MinMax. The other is the receiver-based GBCA protocol, which uses game theory in order to minimize the total network interference. We implemented both protocols in a distributed way, using the Castalia simulator.

Both protocols are evaluated in terms of network performance and compared with a centralized Greedy approach. It is confirmed that multi-channel allocation improves network performance-related metrics, such as packet delivery ratio. However, there are certain algorithm requirements, such as the conflict graph formation, that may not be easily met in many practical scenarios: when the wireless channel's conditions change, conflict graphs have to be reconstructed and thus significant overhead is posed to the network.

Acknowledgements

First of all, i would like to thank my parents and my sister for their love and endless support.

I would also specially thank my thesis supervisor, Assistant Professor Aggelos Bletsas, for his encouragement and constant guidance. His logical way of thinking as well as his accurate remarks helped me a lot to deeply understand many issues that were a bit messed in my mind.

Finally, many thanks to all my adorable friends for their support and the unforgettable moments that we have lived together.

"In theory, theory and practice are the same.
In practice they are not."
- Albert Einstein

Table of Contents

Table of Contents 5

List of Figures 7

List of Tables 10

List of Abbreviations 11

1 Introduction 13
- 1.1 Motivation 13
- 1.2 Related Work 14
- 1.3 Thesis Contribution 15
- 1.4 System Model and Assumptions 16
- 1.5 Outline 17

2 Graphs in practice 19
- 2.1 Link Scheduling 20
- 2.2 IC graph 24
- 2.3 Conflict Graphs 30
  - 2.3.1 Receiver-based 31
  - 2.3.2 Link-based 33
- 2.4 Discussion 35

3 MinMax Channel Allocation 37
- 3.1 Problem Formulation 37
- 3.2 The MinMax Algorithm 38

4 Game Based Channel Allocation 43
4.1 Problem Formulation 43
4.2 The GBCA Algorithm 46

5 Simulation Model and Results 51
5.1 The Castalia Simulator 51
5.2 Simulation Model 55
5.3 MinMax Evaluation 58
5.4 GBCA Evaluation 62
5.5 Comparison 69

6 Conclusion and Future Work 77

Bibliography 80

List of Figures

1.1 IC graph $G = (V, E)$ of a 2-hop tree routing network : the solid lines are *communication* links while the dashed ones represent the *interference* links 17

2.1 A simple tree-routing network graph where arrows represent the tree-links. 21
2.2 The link-scheduling conflict graph for the network of Figure 2.1 with $\chi(G_S) = 3$. 22
2.3 The vertex colored G_S of Figure 2.2. 23
2.4 A time slot assignment for the network of Figure 2.1 with frame length = 4. Notice that $\chi(G_S) = 3$ but $\Delta_S + 1 = 4$. . . 24
2.5 Step 1: Node f records its potential interference links by sensing each PD packet transmission. 26
2.6 Step 2: Node f distinguishes the potential interference links according to the time slot included in a PD packet. 27
2.7 Step 3: Node f uses the SINR model to determine which of the potential interference links actually interfere with a child's transmission. 29
2.8 A deeper look to the IC graph of Figure 1.1 after the *IC graph construction* phase. Assume for simplicity, that no time scheduling information is included. Note also that each interference link disturbs only the communication link marked with the same symbol. 30
2.9 Receiver-based conflict graph, G_R, for the IC graph of Figure 2.8. 31
2.10 Link-based conflict graph, G_L, for the IC graph of Figure 2.8. 35

3.1 Interference links that determine the $C(u, f)$ value for a sender node u on a channel f. 38
3.2 The IC graph of Figure 2.8 and each node's conflict for a single channel in a MinMax channel allocation approach. 39
3.3 An interference-free link-based MinMax channel allocation for the IC graph of Figure 2.8. 41

4.1 The IC graph of Figure 2.8 and payoff values for each player when the same strategy is used in the game based channel allocation approach. 46
4.2 An interference-free receiver-based channel allocation for the IC graph of Figure 2.8 under the GBCA algorithm. 49

5.1 Castalia's basic module structure 52
5.2 Castalia's node's module structure 52
5.3 The 4-hop topology used in our experiments. 58
5.4 MinMax: Iterations to converge vs. Available channels 60
5.5 MinMax: Residual interference vs. Available channels 61
5.6 MinMax: Residual interference ratio vs. Available channels . . 62
5.7 MinMax: Packet Delivery Ratio vs. Available channels 63
5.8 MinMax: Throughput vs. Available channels 63
5.9 MinMax: Latency vs. Available channels 64
5.10 GBCA: Iterations to converge vs. Available channels 65
5.11 GBCA: Residual interference vs. Available channels 66
5.12 GBCA: Residual interference ratio vs. Available channels . . . 67
5.13 GBCA: Packet Delivery Ratio vs. Available channels 67
5.14 GBCA: Throughput vs. Available channels 68
5.15 GBCA: Latency vs. Available channels 68
5.16 Comparison: Iterations to converge vs. Available channels, $\theta = 10$ dB. 71
5.17 Comparison: Iterations to converge vs. Available channels, $\theta = 15$ dB. 71
5.18 Comparison: Packet Delivery Ratio vs. Available channels, $\theta = 10$ dB. 72

5.19 Comparison: Packet Delivery Ratio vs. Available channels, $\theta = 15$ dB. 72
5.20 Comparison: Throughput vs. Available channels, $\theta = 10$ dB. 73
5.21 Comparison: Throughput vs. Available channels, $\theta = 15$ dB. 73
5.22 Comparison: Maximum Latency vs. Available channels, $\theta = 10$ dB. 74
5.23 Comparison: Maximum Latency vs. Available channels, $\theta = 15$ dB. 75
5.24 Comparison: Average Latency vs. Available channels, $\theta = 10$ dB. 75
5.25 Comparison: Average Latency vs. Available channels, $\theta = 15$ dB. 76

List of Tables

5.1 Radio parameter values in Castalia 54
5.2 Wireless channel parameter values in Castalia 55
5.3 Simulation parameter values in Castalia 59

List of Abbreviations

WSN	Wireless Sensor Network
MAC	Medium Access Control
TDMA	Time Division Multiple Access
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
GBCA	Game Based Channel Allocation
MinMax	Minimizing Maximum
PDR	Packet Delivery Ratio
SINR	Signal to Interference-plus-Noise Ratio
RSSI	Received Signal Strength Indication
CBR	Constant Bitrate
RID	Radio Interference Detection

Chapter 1

Introduction

1.1 Motivation

The low-cost, low-power and multi-functional characteristics of wireless sensor networks make them very popular nowadays for a wide range of applications. As the wireless sensor network becomes larger and denser, the use of a single frequency among its nodes may cause severe interference problems and dramatic network performance degradation. That is, when two (or more nodes) which transmit simultaneously can be heard by another node then serious packet reception problems can be created at the latter. In the general case, such simultaneous transmissions must be done on a different time slot or on a different frequency.

The half-duplex radio tranceiver of current WSN hardware can operate on multiple frequencies. Thus, the idea of effectively using them in order to improve network's performance sounds quite attractive. In addition to that, the fact that the number of available channels is limited in practice [3] makes the channel allocation problem very challenging.

Existing multi-channel allocation approaches consist of either *centralized* or *distributed* protocols. The centralized ones require knowledge of the whole network's topology whereas the distributed ones need only local knowledge from each node's neighbors. The distributed protocols are more adaptive to network changes and as a result are more suitable for WSNs.

Furthermore, channel allocation schemes can be classified into *receiver-based* and *link-based* ones. In a receiver-based scheme each node is assigned a specific frequency for packet reception. In a link-based scheme every link is assigned a specific channel for transmission along that link. Link-based allocation schemes allow better spatial reuse [1] but can be only used along

TDMA protocols. That is because every receiver node must know when an expected sender will transmit in order to switch to the sender's channel. On the other hand, receiver-based schemes can be used under both CSMA and TDMA protocols.

The multi-channel allocation problem for an arbitrary network has been proven to be NP-hard [1, 2, 4]. Our goal is to study the fundamental principles of this problem as well as to investigate the assumptions made by existing distributed protocols.

1.2 Related Work

The *centralized* algorithms of [4] statically assign channels to communication links in a heuristic way: a node is assigned the first available frequency in non-increasing order of degrees. The goal is to maximize data aggregation rate at a given sink by minimizing the schedule length, which has been proved to be NP-hard. For that reason, TDMA scheduling and receiver-based multiple frequency assignment are employed. The *centralized* Tree-based Multi-Channel Protocol (TMCP) proposed in [3] statically divides the network into multiple subtrees, allocates different channels to each subtree in a receiver-centric way and forwards data flows only along its corresponding subtree.

The work presented in [2] formulates channel asignment in WSNs as an optimization problem which is proved to be NP-hard. Due to the fact that the number of available channels is limited in practice [3], the problem becomes to optimally assign the limited available channels so as to minimize the total network's interference. The *distributed* receiver-based protocol proposed is called GBCA and adopts a game theoretic approach in order to minimize the total interference suffered by all the receiver nodes of the network. A CSMA/CA MAC is also used for data packets transmission.

Another *distributed* channel allocation algorithm is presented in [1]. In this work a link-based channel allocation protocol is proposed for minimizing maximum interference (MinMax) experienced by any link in the network. This protocol is complemented with a distributed link scheduling protocol which eliminates the remaining interference links in the network by creating

a conflict-free TDMA schedule.

Based on the fact that the *packet reception ratio* (PRR) of a link is highly related to its SINR, the work in [7] addresses the maximization of the PRR of a link by jointly assigning transmission power levels and channels to the sensor nodes of the network. Two heuristic algorithms, both a centralized and a distributed one are proposed to practically solve the above problem.

The works in [2–4] assume that a node is disturbed by another node's transmission if the first is within the interference range of the second. This assumption is quite simplified because the interference range is not always spherical in practice, as shown in [6]. In addition to that, [4] assumes that a node's interference range is equal to its transmission range. On the other hand [1, 7] adopt a SINR-based interference model which is more accurate and use it for the determination of interference links [1] and the PRR value [7] respectively.

1.3 Thesis Contribution

At this work, two already known distributed protocols for channel allocation are examined in a practical way. For this reason the Castalia simulator [12] was used in order to implement the algorithms in a distributed way. Moreover, proposals for creating conflict graphs (a special type of graphs that will be defined in Chapter 2) in practice are considered. A TDMA scheme is employed for data packets transmission. Hence, the idea of using this scheduling information before the channel assignment phase results in a more effective determination of interference links as it will be described in Chapter 2.

The first implemented protocol is the MinMax protocol of [1]. In addition to what has been done in [1] we evaluate the protocol in terms of packet delivery ratio, throughput and for higher receiver sensitivities (i.e. SINR thresholds) as well.

The state-of-the art GBCA protocol [2] was implemented under the realistic SINR interference model. Hence, the payoff function has been changed in the way described in Chapter 4.

1.4 System Model and Assumptions

Firstly it is assumed that the WSN has a *static tree-routing* structure: each child node has only one parent to whom it delivers data messages. There is also a node who serves as a sink (s) and collects the aggregated data packets of the network. Only leaf-nodes generate packets. The non-leaf nodes are used as relays and forward data packets from the leaves to the sink. All nodes transmit data packets using the same power which is called *normal tx_power* and is equal to -10 dBm. Another transmission power, called *max tx_power* and is equal to 0 dBm, is used for transmission of other type of packets. [1]

The network is modeled as an *Interference-Communication* (IC) graph which is also used in [1] and first introduced in [5]. In an IC graph $G = (V, E)$, V denotes the set of all nodes of the network (including sink) and E the set of all edges $e = (u, v) \in E$. Each edge e represents a link from node u to node v which can be either a *communication* or an *interference* link.

A communication link will exist, if node v's RSSI exceeds the threshold of -87 dBm when u is transmitting. According to the experiments done in [8], if an RSSI value is over that threshold, then high packet reception rates of at least 85% will be achieved. So every link between a parent and a child node in the routing tree (also called *tree-link*) is a communication link. Note here that only a subset of communication links forms the routing tree. Every communication link is also assumed to be *symmetric* i.e. if node v can receive packet from u, then u can receive from v as well.

An interfering link $e = (u, v)$ indicates that u's transmission interferes with any transmission intended for v, even though u's packet may not be correctly received by v.[2] An axample of an IC graph can be shown in Figure 1.1: Node f has two children (nodes d, e) and an incoming interference link (b, f) from b. Note that the transmissions of a and d can be overheard by the sink (s) while c can hear only its two children a and b.[3] If E_T denotes

[1] Such as packets that are used during different phases of the implemented algorithms. Section 2.2 explains why max tx_power is needed.

[2] Since a static tree-routing is adopted, the transmissions intended for v are the transmissions of its children.

[3] A node can hear another node's transmission if the power level sensed at the receiver exceeds the receiver's sensitivity value.

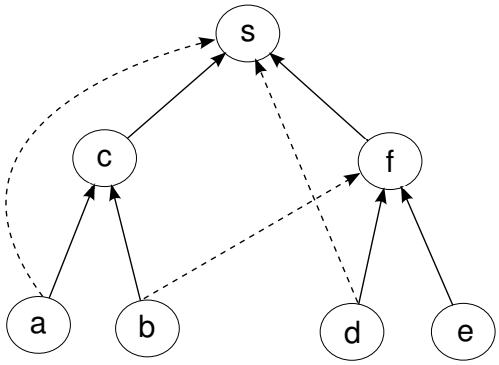

Figure 1.1: IC graph $G = (V, E)$ of a 2-hop tree routing network : the solid lines are *communication* links while the dashed ones represent the *interference* links

the tree links, then $E_I = E - E_T$ is the set of all interfering links in the IC graph. Moreover, interference is modeled under the SINR model and will be discussed in Section 2.2.

Another assumption is that each node has a *single tranceiver* radio and antenna and cannot receive messages from multiple senders at the same time. This tranceiver is also assumed to be *half-duplex* which means that each node can either transmit or receive packets at a specific time slot. The available frequencies are non-overlapping, non-adjacent and they do not interfere with each other. As shown in experiments in [3], adjacent channel interference may cause serious collision and packet reception problems in a WSN.

1.5 Outline

The next chapters of this thesis address the following issues:

Chapter 2 describes a practical way for creating an IC graph using the SINR model. The concept of conflict graphs is presented and a functional way of extracting them from an IC graph is introduced. A simple link-scheduling protocol will be also presented.

In Chapter 3 the problem of channel allocation will be examined with a link-based approach and the implementation of the MinMax protocol in

Castalia simulator. Chapter 4 exhibits the game theoretic approach of the channel allocation problem [2]; the GBCA algorithm is also analyzed.

The Chapter 5, firstly introduces the basic characteristics of the Castalia Simulator as well as its setup for our experiments. Then, a centralized greedy channel allocation algorithm created in Matlab is presented and compared with the other two distributed algorithms. Numerical results are offered and discussed. Finally, conclusion is provided at Chapter 6.

Chapter 2

Graphs in practice

In a channel allocation problem the first and very important issue for a node is to determine its *interference* links. That is, a receiver node v has to learn the nodes whose transmissions interfere with the transmissions intended for v.[1] For this reason node v should have a priori *scheduling* and *routing knowledge* for both communication and interference links. Each node needs routing information in order to distinguish between the links that can be heard by its radio: whether they come from a child or any other sender node. As a result each receiver node must know the IDs of its own children.

The term scheduling refers to "when" two nodes are about to transmit a packet. Every receiver node v needs this information in order to check if a transmission along a non-tree link is going to happen at the same time with a child's transmission. [2] If that happens, v has to decide whether the non-tree link heard interferes with v's reception from a child. This decision is taken by v using the SINR interference model to calculate its SINR value for the specific concurent transmissions. If this SINR value is below a minimum threshold then the heard non-tree link is determined as *interference* link. Otherwise it is ignored. In case that an overheard by v non-tree link is activated at a different time slot from any of v's children, then children's transmissions are assumed conflict-free.

At this point questions like "*How does a node get such knowledge ?*" or "*Is a node able to learn all this information in a distributed way ?* " seem reasonable. If reliable scommunication between the sender and the receiver node of an interference link can be ensured, then the answer to the second

[1] In the link-based approach that will be examined the ID of such nodes is needed while in the receiver-based one the ID of their parent has to be learnt.

[2] We call as non-tree link of v every link that can be heard at v and has none of v's children as sender.

question is definitely "yes". In section 2.1 a distributed protocol is presented, used for acquiring all necessary scheduling information. In section 2.2 the determination of interference links will be discussed in a detailed way. Section 2.3 introduces the concept of conflict graphs: A special kind of graphs which can be extracted from a given IC graph.

2.1 Link Scheduling

Each WSN device is equipped with one half-duplex radio. This imposes two main communication costraints: 1) Each node cannnot transmit and receive simultaneously and 2) each node cannot receive from more that one senders at the same time. Therefore, each node's children links are considered as *conflicting* and must be activated on different time slots. Additionally, a child must not transmit when its parent is transmitting too.

In order to overcome these hardware limitations, conflicting transmissions must be done on different time slots. A simple distributed link-scheduling protocol, like the one described in [1], for time slot assignment that minimizes the frame length is presented in this section. The result is a TDMA schedule, where each node activates its tree-link at the chosen time slot in every frame. In TDMA a sequence of time slots form a frame. The duration of a time slot equals to a fixed-length packet's transmission time. The maximum chosen number of time slots indicates the frame's length. The first part of the link scheduling protocol is the creation of a graph called *link scheduling graph*. Each node communicates then with its neighbors in the graph and chooses a time slot according to a simple deterministic algorithm.

Given a network's graph $G = (V, E_T)$, [3] a schedule conflict graph denoted by $G_S = (V - \{s\}, E_S)$ is constructed in the following way:
Every node of G, except from the sink, participates also in G_S. An edge $e \in E_S$ will be created between two nodes if these nodes are siblings in G i.e. they have the same parent or a parent-child link exists between them.

Consider the tree-routing network with the parent-children links (tree

[3] It can also be considered as an IC graph with no interference links

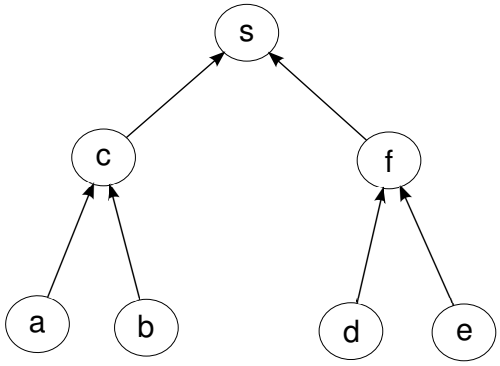

Figure 2.1: A simple tree-routing network graph where arrows represent the tree-links.

links) of Figure 2.1. According to the limitations discussed, sibling nodes *a, b* should not transmit at the same time. Similarly *d* with *e*, and *c* with *f* must transmit on different time slots. Of course *a* and *c* or *b* and *c*, must not send packets simultaneously (the same for *d* and *f* or *e* and *f*). The link scheduling conflict graph can be shown in Figure 2.2. **The idea is that two neighbor nodes in G_S must be assigned different time slots**. Considering every *time slot* as a different *color*, vertex coloring of G_S provides the solution needed: minimize the number of colors (time slots) needed for G_S such that no edge connects two identically colored vertices. The minimum number of colors required is called the *chromatic number* and is denoted $\chi(G_S)$. .

In practice each node (apart from the sink) has to do the following:

1. Firstly, each node broadcasts consecutively a *Time_Slot_Discovery packet* which contains its own ID and its parent's ID as well.[4]

2. Every node checks whether the sender of a received *Time_Slot_Discovery packet* is any of its children, or if the contained parent ID equals to its own parent ID (sibling nodes).[5] If so, then the sender's ID is kept in a

[4]With the term consecutively it is implied that two nodes shoud not transmit simultaneously so as transmission conflicts are avoided. In fact each node delays its transmissions for a time period corresponding to its ID.

[5]Each node learns its children in a *Children Discovery phase* that will be explained in section 2.2

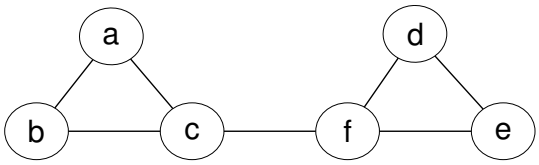

Figure 2.2: The link-scheduling conflict graph for the network of Figure 2.1 with $\chi(G_S) = 3$.

Link_Scheduling_Neighbors list. By the end of this step, every node will have determined its neighbors and thus G_S will have been created.[6]

3. Now nodes have to communicate with their neighbors in G_S. The idea of the link scheduling algorithm is that every node broadcasts (when its turn comes) a *Time_Slot packet* with its ID and a chosen time slot. The chosen time slot is the current smallest available slot at that node.[7] Note that a slot is not available if it has been chosen by a neighbor. Note also that two neighbor nodes are not allowed to choose a time slot at the same time and therefore priority is given to the nodes with the smallest IDs.

4. At this step every node has chosen a different slot from its neighbors in G_S. Now each node has to learn the maximum value slot of the assignment (i.e. the frame length). Note that the node which has chosen the largest slot may be more that 1-hop away. Thus, each node turns to *max tx_power* and broadcasts a *Max_Slot_Discovery packet* with its ID and the number of the chosen time slot. Each node rebroadcasts then every *Max_Slot_Discovery packet* received. In this way the Max_Slot_Discovery packets are flooded in at least 2-hop range.

Lets explain how the above procedure is performed for the network of Figure 2.1. Steps $1-2$ create the G_S shown in Figure 2.2. According to step

[6]Note that two neighbors in a G_S may be more than 1-hop away. In this case communicaton is achieved either at maximum tx_power or using the concept of *indirect communication* that will be explained in section 2.3.

[7]For example time slot 1 is chosen by every node that has the smallest id from all of its neighbors in G_S.

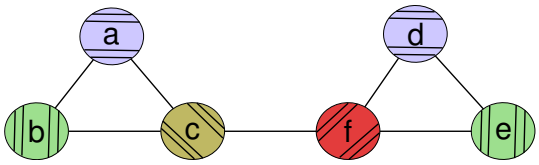

Figure 2.3: The vertex colored G_S of Figure 2.2.

3. node *a* chooses *time slot 1* and broadcasts a *Time_Slot packet*. Node *b* excludes *time slot 1*, chooses *time slot 2* and broadcasts its own *Time_Slot packet*. In the same way *c* chooses *time slot 3*. It has to be noted that node *d* has not received a *Time_Slot packet* from its neighbors and thus its smallest available slot is *time slot 1*. Node *f* is the latter node who chooses slot (it has the largest ID). So *f* has already excluded the first three available slots (cause they are occupied by its neighbors) and thus chooses *time slot 4*. The vertex-colored graph G_S created at this step is depicted in Figure 2.3. We can see that node *a* has the same color with node *d* and node *b* has the same color with *e* as well. This happens because neither *a* is neighbor with *d* nor *b* is neighbor with node *e* in G_S. It can be also seen that *c* and *f* have been assigned different colors. The frame of slots will be repeated after the last slot. Hence, the frame length is equal to the largest assigned time slot. At the end of step 4. each node (with sink included) will know that the frame length = 4. The already described assignment can be shown in the graph of Figure 2.4. The number in each rectangle denotes the time slot that each node is allowed to transmit a packet.

Theorem 2.1. *The maximum frame length determined by the link scheduling algorithm is* $\Delta_S + 1$*, where* Δ_S *is the maximum degree of* G_S*.*

Proof. Assume that time slots are numbered in increasing order *(1,2,3...)*. Every node $u \in V - \{s\}$ of G_S must choose the smallest available time slot that is different from its neighbors. Let m the node with the maximum degree Δ_S. If all m's neighbors choose time slots, then the first Δ_S slots will be occupied. As a result the largest possible slot that node m can choose is the $\Delta_S + 1$ slot. □

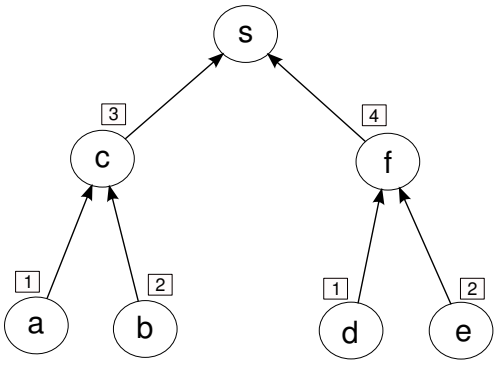

Figure 2.4: A time slot assignment for the network of Figure 2.1 with frame length = 4. Notice that $\chi(G_S) = 3$ but $\Delta_S + 1 = 4$.

Notice that for the specific topology the minimum number of time slots required is $\chi(G_S) = 3$ but the maximum frame length determined is $\Delta_S + 1 = 4$. It must be also noted that material for this section was first published in [1].

2.2 IC Graph

For the construction of the IC graph in our approach, a node should have learnt the necessary routing and scheduling information that we have already discussed at the beginning of this chapter. So a *children discovery* and a *link scheduling phase* precede the *IC graph construction* phase. The static routing network models that we examine assume that each node knows a priori its parent node's ID. This is the *minimum* required knowledge for starting the presented algorithms.

A node can discover its children in the simple following way: Every node broadcasts in *normal tx_power* a *Children Discovery packet* which includes its ID and its parent's ID. The nodes that receive this packet, store the ID of the packet's sender in a children's list if the parent ID of the packet equals their own ID. The link scheduling phase follows the children discovery phase. As it was described in the previous section 2.1, each node chooses a time slot where it is allowed to transmit.

Now the network's IC graph has to be created and stored in a distributed

way. A practical solution for constructing it is presented in RID protocol [6]. Zhou et al. have observed in experiments that a node's communication range is not equal to its interference range: In case of a weak link, the received signal's power level is low and thus, it can be easily get interfered by one or more distant jammers. So a packet may not be correctly received and in this case the interference range is presented as smaller than the communication range. On the contrary, when a strong link exists, the communication range is presented as bigger than the interference range because it is hard to disrupt a strong signal. The basic concept of the proposed RID protocol [6] is that each node keeps the power levels of the nodes it can hear and uses them to figure out all the possible collision cases in the network. This idea was adopted in our simulations in order to allow each node to calculate its interference links.

In the weak link's case it is implied that a node's packet reception may be disrupted by one or more weak signals.[8] Each receiver node has to learn both the ID of the sender node of such signal and its power level. So a simple solution could be to let each node broadcast a packet with its ID, while the other nodes sense this transmission. However, a packet from a weak link may not be correctly received and thus its ID cannot be learnt. The idea is that each node must broadcast a *sequence of two packets* as follows:

The first packet is a *notification packet* that contains its ID and is transmitted in *max tx_power* so as reception probability through a weak link is maximized.[9] Then the transmitter node waits untill the hardware is ready to send again and transmits a *power detection* packet (PD packet) in normal tx_power.[10] In this way when a receiver node gets a notification packet, automatically learns who is about to send a PD packet and starts sensing its transmission.

A struct (ID, power_level) is created at each node with the sensed power level and the corresponding sender's ID for each PD packet. Each node maintains a list of such structs called *Potential_Interference_In* list. Structs with ids that belong to either of node's children or node's parent are not

[8] It is assumed that all nodes transmit in normal tx_power.

[9] It also includes the time slot chosen at the link scheduling phase and the ID of its parent. The usefulness of these infos will be explained in the following paragraphs.

[10] This time is equal to 25 ms in our simulations.

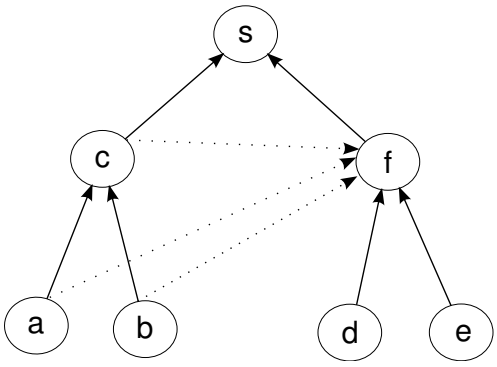

Figure 2.5: Step 1: Node f records its potential interference links by sensing each PD packet transmission.

inserted in this list. That is because if the constraints presented in section 2.1 are satisfied (i.e. two-hop neighboring nodes in the routing tree do not transmit simultaneously), then transmissions from such nodes cannot cause interference to the node that owns the list. As a result, a separate list of structs with children IDs called *Children_In* list is kept whereas PD packets from parent nodes are being ignored. The IDs in Potential_Interference_In list represent nodes who *potentially* cause interference to the owner-node of the list. The term "potentially" is used because a power level may not be high enough to interfere with an owner's node reception from a child (strong link's case).

In Figure 2.5 we can see the PD packets transmissions that node f can sense. That is, node f apart from its parent and children can also hear nodes c, a, b. As a next step, node f uses the a apriori scheduling knowledge in order to distinguish between the heared non-tree links. As it is illustrated in Figure 2.6, node f ignores the link (c, f) since it will be scheduled at time slot 3. Note that at this time slot neither of f's children will transmit and thus no reception problems will be caused at f.

In order to distinguish which of the potentially intefering senders do actually interfere with the communication from a child, each node uses the SINR model to calculate the interference suffered by such links. Consider a non-leaf node v, $C = \{c_1, c_2, ..., c_n\}$ a set of v's children nodes and $J = \{k_1, k_2, ..., k_m\}$

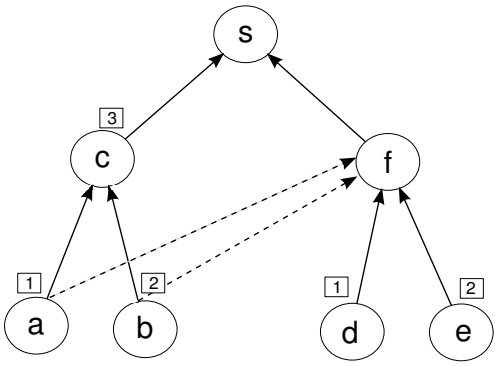

Figure 2.6: Step 2: Node f distinguishes the potential interference links according to the time slot included in a PD packet.

a set of jammer nodes. For a node v a jammer node $k_i \notin C$ is assumed to be a node that transmits at the same time slot with either of v's children and can be heard by v. Note here that a child's and a jammer's time slots are known to v since each node includes this info in the notification packet transmitted. The SINR at v for any child-parent link, (c_i, v) with $c_i \in C$ is given by:

$$SINR_{(c_i,v)} = \frac{r_{(c_i,v)}}{\sum_{j=1}^{m} r_{(k_j,v)} + N_v} \tag{2.1}$$

where $r_{(c_i,v)}$ is the received power at node v when its child c_i is transmitting a PD packet, $r_{(k_j,v)}$ is the received power at node v when the jammer node $k_j \in J_v$ is transmitting a PD packet and N_v is the power level of the white backround noise around v. Note that $r_{(c_i,v)}$ belongs to a struct of *Children_In* list while $r_{(k_j,v)}$ has been stored in a struct of *Potential_Interference_In* list. If the calculated SINR is below a predefined *threshold* value then the respective jammer nodes k_j are determined as *interfering nodes*.

It is very important to point out that if a jammer node k_1 of v cannot interfere with a communication link in v this does not mean that the composition of k_1 and another jammer node's k_2 will not interfere too. In other words the composition of multiple negligible jammers is not necessary negligible [6]. Behind this observation it is implied that each receiver node v has to calculate the SINR for all possible combinations of jammers and children

that exist in the corresponding lists. Additionally, these calculations can be done considering one, two or more jammers simultaneously.

In our simulations it is assumed that there are at most two jammer nodes ($m = 2$) while a child transmits. [11] Hence each node v performs the following procedure:

- Firstly, node v calculates SINR for $m = 1$, for every child $c_i \in C$ and for every $k_1 \in J_v$. So equation 2.1 becomes:

$$SINR_{(c_i,v)} = \frac{r_{(c_i,v)}}{r_{(k_1,v)} + N_v}$$

  We denote θ the SINR value in dB that will be used as a threshold for the determination of interference links. Thus, if $SINR_{(c_i,v)} < \theta$ then transmissions of c_i and k_1 are determined as conflicting.[12] A struct (c_i, k_1) is created for each conflict pair and stored in a *Interference_Per_Two* list.

- For $m = 2$ the SINR is calculated for every child $c_i \in C$ and for the $k_1, k_2 \in J$ that do not interfere individually. That is k_1, k_2 must not exist in any struct of Interference_Per_Two list. Equation 2.1 becomes:

$$SINR_{(c_i,v)} = \frac{r_{(c_i,v)}}{r_{(k_1,v)} + r_{(k_2,v)} + N_v}$$

  As in the previous case, if $SINR_{(c_i,v)} < \theta$ then c_i, k_1 and k_2 are set as conflicting transmitters. This information is kept in a *Interference_Per_Three* list which has structs of the form (c_i, k_1, k_2).

After the SINR calculation, each node has determined the interference links that disturb a child's transmission. For example, in Figure 2.7 it can be shown that node f has determined (b, f) as interference link. This means that, the SINR value of f when b and e transmit falls behind the predefined

[11] In more dense networks than the ones examined in this thesis more jammer nodes should be considered.

[12] The thresholds, θ, of 10 and 15 dB were used in our simulations.

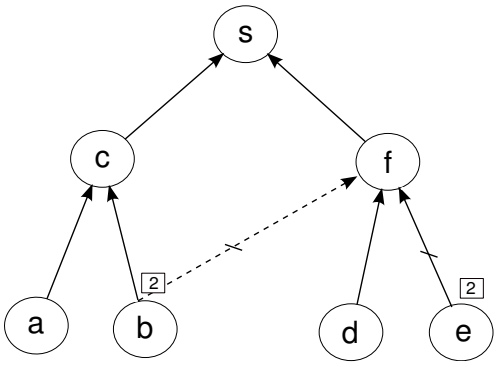

Figure 2.7: Step 3: Node f uses the SINR model to determine which of the potential interference links actually interfere with a child's transmission.

threshold. On the contrary, $SINR_{(d,f)}$ when a transmits is greater than θ and thus the potential interference link (a, f) (shown in Figure 2.6) is ignored.

With the end of the above procedure the IC graph construction phase comes to its end. Now a great part of the IC graph of the network has been stored in a distributed way: each node has knowledge of its tree-links (incoming and outcoming) as well as of its incoming interference links, i.e. the nodes that may interfere with its reception from any child.

The remaining information that must be learnt is the outgoing interference links that each node may have, i.e. the nodes that a sender node, u may interfere. This is done during *conflict graph creation* phase described in the next section, where every non-leaf node broadcasts to all nodes in the IC graph an *Outgoing_Link_Discovery packet* which includes its ID and a table called *Interference_Senders*, with the stored sender IDs for all the calculated interference links.[13] If u receives such a packet and finds its own ID in the table then u gets to know that it has an interference link on the transmitter of the packet.

It has to be mentioned that the SINR calculation complexity is reduced to a great extent by the fact that scheduling information of the heared links (i.e the time slots at which they are being activated) is taken a priori into

[13]All k_1 values in *Interference_Per_Two* list and all k_1, k_2 values in *Interference_Per_Three* list are broadcasted in max tx_power so as reception probability through interference links is maximized.

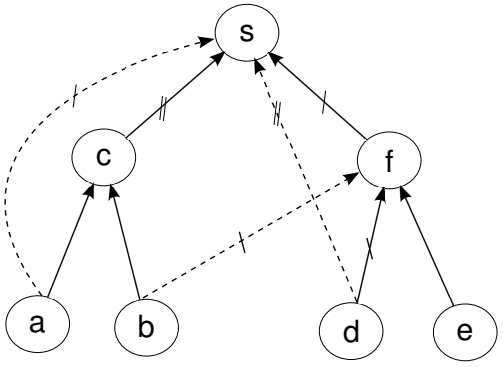

Figure 2.8: A deeper look to the IC graph of Figure 1.1 after the *IC graph construction* phase. Assume for simplicity, that no time scheduling information is included. Note also that each interference link disturbs only the communication link marked with the same symbol.

account: calculations between links with different time slots are never been done. Besides, potential interference links that are scheduled on different time slots are not considered and thus more efficient channel allocation can be achieved and with fewer channels as well.

Last but not least, a major drawback of the approach discussed in this section is that when the wireless channel characheristics change (due to channel fading for example) then the stored power_level values have to be changed too.[14] In such cases IC graph construction phase must be revoked and thus significant overhead is posed to the network.

2.3 Conflict Graphs

A special type of graph called *conflict graph* has to be extracted now from the IC graph. This graph includes the interfering nodes of a network and the connections between them as well. Note that the concept of conflict graph formation is the same with the one used in [1].
Two main channel allocation approaches are examined in our work: *receiver-*

[14]The wireless channel model used in our experiments does not include fading so if the tx_power of a node A is fixed, then the received power level at node B is fixed during the whole simulation.

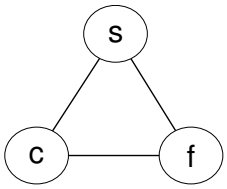

Figure 2.9: Receiver-based conflict graph, G_R, for the IC graph of Figure 2.8.

based channel allocation and *link-based* channel allocation. In receiver-based channel allocation, each node is assigned a fixed channel for receiving messages. Thus, children nodes have to use their parent's channel to send him a message. In a link-based channel allocation every sender node is assigned a channel which uses to send packets along a certain link [1]. This needs that a parent node switches appropriately every time to the transmitting child's channel.

2.3.1 Receiver-based conflict graph

In a receiver-based channel allocation, two *receivers* are named as *interfering* if the transmission of any child of one receiver is interfered by the transmission of a child of the other receiver [1]. The goal is that every receiver v must be assigned a channel that is different from all of its interfering receivers' channels. If this is possible then all network's interference is eliminated.

For this reason a new graph called *receiver-based conflict-graph* is extracted from a given IC graph. A receiver-based conflict graph $G_R = (R, E_R)$, is a graph where R is the set of all receiver nodes of IC graph (including sink). An edge $e \in E_R$ is created between two interfering receivers. Assume the IC graph of Figure 2.8. The actual transmission conflict pairs are illustrated: The interference link (b, f) interferes in practice only with the transmission link (d, f). Similarly, (a, s) interferes with (f, s) at s and the link (d, s) conflicts disturbs the transmission of (c, s).

In Figure 2.9 we can see the receiver-based conflict graph for the IC graph of Figure 2.8. G_R consists of only the receiver nodes c, f and s. Node c ic neighbor with node s (there is an edge between them) because the child node

a of c has an outgoing interference link to s. In the same way b's transmission interferes with a reception in f and so c is conflict neighbor with f. Node s has also an incoming interference link from d that's why f is neighbor with s in G_R.

Consider that every channel is a different color. For an interference-free channel allocation, each node in G_R must be assigned a channel that is different from all of its interfering receivers' channels. Thus, vertex coloring of G_R minimizes the number of colors (channels) needed for G_R such that no edge connects two identically colored vertices.

It must be noted that all edges in G_R are undirected. This means that each node should be able to communicate with all of its neighbors in G_R. However two nodes in a G_R may not be one-hop away from each other in an IC graph. For example node c and node f in Figure 2.8 are two-hops away. At first thought, there are two solutions that can be used: 1) Each node could increase its transmission power or 2) each node could use its default sending power and relay the information among multiple hops. The first option may not suffice in some cases.[15] The second option requires a reliable end-to-end route which is in contradiction with the initially assumed network model: "A reliable route has to be ensured in an unreliable network!".[16]

The idea is that each receiver node v (in R) communicates with a conflict neighbor in an *indirect way*. This solution consists of two succesive broadcasts in *max tx_power*. Lets see again Figure 2.8: When node c for example wants to communicate with its neighbors s and f it switches to max tx_power and broadcasts its packet. Nodes a and b receive such packet and then rebroadcast it using max tx_power too. Finally, nodes s and f receive the initial c's packet through the interference links (a,s), (b,f) respectively. In the same way information from node f is passed to its neighbors through the interference links (f,b), (d,s) and communication link (f,s). Notice that for the specific topology, node s can communicate both directly through the communication links (s,c), (s,f) and indirectly through the interference links

[15] In large networks conflict nodes may be three or more hops away and transmisison power may cannot be increased further.

[16] Since interference has not been eliminated, the network is assumed unreliable.

(s,a), (s,d). Note also that max tx_power is used so as packet reception through interference links becomes feasible.
In order for a receiver-based conflict-graph to be created in practice, the procedure mentioned above has been implemented in the following way:

1. Firstly, each non-leaf node broadcasts consecutively to every node that can hear it in the IC graph an *Outgoing_Link_Discovery packet* using max tx_power.

2. Then, every node that receives such a packet, adds its own ID and rebroadcasts (also in max power) the packet (now the packet contains two IDs, the initial sender's ID and the intermediate sender's ID that rebroadcasts it).

3. When all rebroadcasts have finished and no one transmits in the channel, each receiver node v determines its *interfering receivers* (i.e. its neighbors in G_R) as follows:

   - For every rebroadcast packet that has received by its children, v checks if any child's ID is in the Interference_Senders table. If so, then the initial sender of the packet is determined as an *interfering receiver*. In this way, each node v learns the nodes that are being interfered by v's children.
   - For every interference link (u, v) determined in *IC graph construction* phase, v adds as *interfering receiver* the parent of the sender of the link (u's parent). Note that the parent ID needed is known since it was included in the *notification packet* that u has sent in the beginning of *IC graph construction* phase (section 2.2).

2.3.2 Link-based conflict graph

Two *senders* in *link-based* channel allocation are called *interfering* if one's transmission is interfered by the other's transmission. So in this case every sender must use a channel for transmission that is different from the ones used by its interfering senders [1]. For this case, another graph has to be

extracted from a given IC graph, which is called *link-based conflict-graph* and is denoted by $G_L = (V - \{s\}, E_L)$. Every sender node of the network can be also a node in G_L. An edge $e \in E_L$ exists among two interfering senders.

Figure 2.10 illustrates the link-based conflict graph of the IC graph of Figure 2.8: Node a has as interfering neighbor in G_L the child node f of s. In the same way, node d which has also an outgoing link to s, is neighbor only with c, while node b must communicate in G_L with the node d. Note that the sender node e does not have any neighbors in G_L and this means that it is not needed to switch to a different transmission channel.

Similarly with a receiver-based conflict graph, vertex coloring of G_L (assuming each channel as a different color) results in an interference-free channel allocation. Besides, every edge $\in E_L$ is an undirected edge and the concept for communication between interfering senders is the same with the one adopted for the commmunication in a G_R.
However, the way a link-based conflict graph is created in practice is different than the one described for a G_R:

1. Assume again, R the set of all receiver nodes of the IC graph. Each non-leaf node $v \in R$ broadcasts consecutively to every node that can hear it in the IC graph a *Conflict_Link_Discovery packet* using max tx_power . This packet includes a table called *Interference_Links* table, with all the conflict transmitters sets found in v's *Interference_Per_Two* and *Interference_Per_Three* lists.

2. Every node $n \in V - \{s\}$ that receives a Conflict_Link_Discovery packet determines its *interfering senders* (i.e. its neighbors in G_L) as follows:

   - If n is a child of v, then n checks the *child* field of every conflict set in Interference_Links table. If this field equals n's ID, then *conflict_sender_1* and *conflict_sender_2* (only if it has a non-zero value) are set as *interfering senders*. [17]

[17]Note that each line of the table is a record of the form (*child*, *conflict_sender_1*, *conflict_sender_2*). In case that the line refers to a conflict sender that interferes individually with the respective child (remember the SINR calculation for $m = 1$), then *conflict_sender_2* field is given by default a negative value.

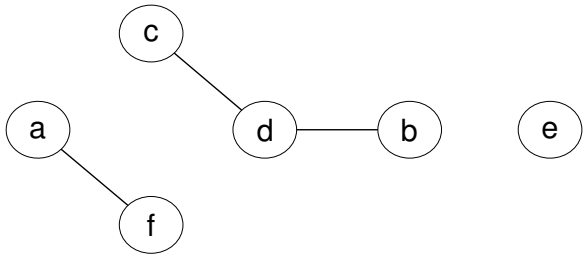

Figure 2.10: Link-based conflict graph, G_L, for the IC graph of Figure 2.8.

- In case that n's ID equals either *conflict_sender_1* or *conflict_sender_2* in any tuple in Interference_Links table (it means then that (n, v) is an interference link at v), then n defines as an *interfering sender*, the node in the corresponding *child* field.

2.4 Discussion

Both types of conflict graphs investigated in this section can be constructed in practice in a distributed way as described. Note that a reception from an interference link is a critical factor for both the graph's creation and the communication between neighbors (recall the concept for indirect communication). If such a reception cannnot be done, then conflict graphs cannot be created. Besides, when channel conditions change, the received power values stored during the IC graph construction phase must be also updated. This means that the IC graph as well as the corresponding conflict graph must be reconstructed from scratch and as a result significant overhead is added to the network.

Simple heuristic algorithms exist for interference-free channel allocation as the ones proposed in [1]. These algorithms require at most $\Delta + 1$ channels, with Δ being the maximum degree of the corresponding conflict graph (the proof is similar to Theorem 2.1). However, in case that the number of required channels exceeds the number of available channels (The number of available non-overlapping channels is limited in practice [3]) there exist

algorithms as the ones that will be examined in Chapter 3 and Chapter 4 that minimize interference using a limited number of available channels.

Chapter 3

MinMax Channel Allocation

When the number of available channels does not suffice to remove all interference in a network, then a fair channel allocation scheme has to be adopted [1]. In this chapter a channel allocation scheme that minimizes the maximum interference experienced by any link of a WSN will be presented. Furthermore, MinMax channel allocation has been proved to be NP_hard [1]. Thus, a distributed algorithm firstly introduced in [1] that performs such allocation in polynomial time will be analyzed.

3.1 Problem Formulation

The idea of this approach is that each node tries to minimize its own conflict in a fair way. The local decisions that each node takes, result in a globally fair channel assignment. Let $S = V - \{s\}$ the set of all sender nodes in a network, $u, w \in S$ and p_u, p_w the parents of u and w respectively. The definition used for a node's conflict is the same as in [1]: A conflict of a transmitter node u (or conflict of the transmission link (u, p_u)) on a channel f is denoted $C(u, f)$ and defined as:

$$C(u, f) = |\{(w \in S) \,|\, (f(u) = f(w)) \ \wedge \ ((w, p_u) \in E_I \ \vee \ (u, p_w) \in E_I)\}|$$

That is, $C(u, f)$ is equal to the number of nodes that use the same transmission channel with u and have

- an outgoing interference link to the parent of u or
- an incoming interference link to their parent from u

In Figure 3.1 we can see that $C(u, f) = 1$. Note that despite the fact that u both causes interference (outgoing link (u, p_w)) and suffers interference

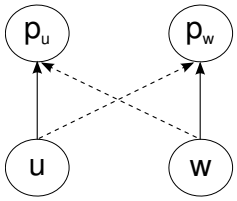

Figure 3.1: Interference links that determine the $C(u, f)$ value for a sender node u on a channel f.

(incoming link (w, p_u) to p_u), the conflict metric (i.e. number of nodes) is considered only once. For a MinMax channel allocation approach of the IC graph of Figure 2.8, the conflicts of each node under a single frequency are illustrated in Figure 3.2. Node a interferes only with the transmission of f at s and as result has $C(a, f_1) = 1$. Similarly node b has also the same conflict value since it interferes with the transmission of d. Moreover, nodes c, f have $C(c, f_1) = 1$ and $C(f, f_1) = 1$ since they suffer interference from d and a respectively. Note that e's transmission is not interfered by any link and thus $C(e, f_1) = 0$. Node d has the maximum conflict value, since its transmission is interfered by the link (b, f) and in the same way d causes interference to the link (c, s).
The MinMax allocation problem that has to be solved can be formulated as:

$$\begin{aligned} &\text{Minimize} \quad \max\{C(u, f), \forall u \in S\} \\ &\text{Subject to} \quad 1 \leqslant f \leqslant |F| \end{aligned}$$

where $|F|$ the available number of non-overlapping channels.

3.2 The MinMax Algorithm

For the already mentioned MinMax allocation problem the minimization of the maximum conflict of a tranmission link is needed. Thus, a link-based channel allocation approach will be adopted since different channels can be assigned to links. For this reason, a link-based conflict graph will be used. Note that each node communicates only with its neighbors in G_L in order to

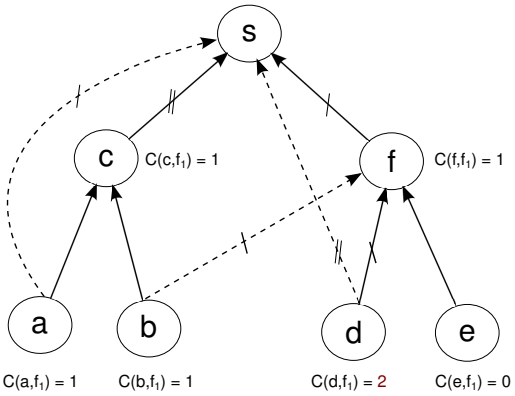

Figure 3.2: The IC graph of Figure 2.8 and each node's conflict for a single channel in a MinMax channel allocation approach.

calculate its conflict. The phases that have preceded the channel allocation phase that will be described are the following:

1. Link-scheduling phase
2. IC graph construction phase
3. Link-based conflict graph creation

Now every node $u \in S$ is ready to run the MinMax channel allocation algorithm, which consists of the following phases:

1. Initialization

Initially each node u chooses a random channel in the range between 1 and $|F|$. Then each node consecutively broadcasts to each neighbors in G_L in *max tx_power* a *Channel_Allocation packet* that includes its ID and the already chosen channel. When all transmissions have finished, every node rebroadcasts (also in max tx_power) only the packets received from either its children or its interference links' senders with its ID added. [1] Then, each node keeps only the rebroadcast packets that have as initial ID the ID of any neighbor in G_L. Note that, this

[1]This is the idea for indirect communication in a conflict graph, explained in 2.3.1. Remember that max tx_power is used so as packet reception through interference links becomes possible.

communication scheme will be used for every packet broadcast in the next phases. Note also that all nodes will be using the same channel for communication.

2. Confict Calculation

At this phase, each node u has knowledge of its neighbors' (in G_L) chosen channels and thus calculates its conflict $C(u, f)$, where f is u's current chosen channel. For every neighbor that has chosen the same channel with u, $C(u, f)$ is incremented by one. A *Current_Conflict packet* is then sent with u's ID and the $C(u, f)$ value.

3. Channel Choice

By the end of Confict Calculation phase, each node will have learnt the current conflits of its neighbors as well as their chosen channels. These two pieces of information are used in the following way:

- Neighbors' channels are used by u in order to calculate $C(u, f)$ for every available channel.
- Initially each node keeps a list with $|F|$ available channels. However, a channel can be excluded in the following case: If node u receives a $C(n, f)$ from a neighbor n in G_L and $C(n, f) > C(u, f)$ then channel f is considered unavailble at u. This is the key point of the algorithm, since it prevents u to switch to a channel that will increase its neighbor's conflict and probably the network's max conflict. Now each node u chooses the channel from the updated available channels list that results in the smallest conflict $C(u, f)$. A *Channel_Allocation packet* is broadcasted then in the way described in phase 1. It has to be noted that two neighbor nodes in G_L must not choose channels simultaneously. For this reason priority is given to the nodes with the smallest ID.

4. Convergence

At this point the convergence of the algorithm is examined. As long as there is a node u that can decrease its $C(u, f)$ with its available

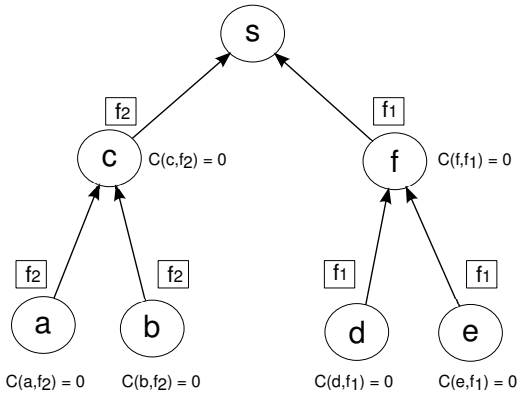

Figure 3.3: An interference-free link-based MinMax channel allocation for the IC graph of Figure 2.8.

channels, then the phases 2-4 will be repeated. In our distributed implementation, if u cannot further decrease its conflict and none of u's neighbors has broadcasted a Channel_Allocation packet in the previous round (note that every neighbor must have broadcasted at least one channel_Allocation packet, then the procedure is terminated for node u. Obviously, the algorithm converges, when the above procedure is terminated at every node in G_L.

When the algorithm converges, each node u must notify its parent node in the IC graph for u's chosen channel. A TDMA scheme will be used and hence each parent node must be aware of its children's transmission channels. Therefore, a *Channel_Notification* packet is sent in *normal tx_power*. This packet includes the node's finally chosen channel as well as its ID. The following theorem proves the convergence of the MinMax algorithm [1].

Theorem 3.1. *The MinMax algorithm converges in at most $|E_I|$ rounds, where $|E_I|$ is the total number of interference links in the IC graph, G.*

Proof. In every round, at least one node u chooses a channel. Since two neighbors in a G_L cannot choose channels at the same time, u's neighbors

keep their channels unchanged for the current round. The channel that node u chooses leads to a decrease of its own $C(u, f)$ in the current round. Hence, the total number of interference links between u and its neighbors decreases, which implies that at least one interference link is removed in G. So, at the worst case that one interference link is removed in every round, the algorithm needs at most $|E_I|$ rounds to converge. □

When there are enough available channels or equivalently in sparse networks as the one of Figure 3.2, the MinMax algorithm results in an interference-free allocation as it is illustrated in Figure 3.3. We can see that all conflicts have been eliminated and each node uses the frequency inside each rectangle to transmit a packet. Assume that all nodes use the same frequency f_1 at start. The nodes with the smallest ID in each neighborhood choose channels first. Thus, according to the G_L of Figure 2.10 nodes a, b, c choose the first available channel that is different from their conflict neighbors' and as a result they switch to frequency f_2. In this way, nodes' f, d conflicts are also eliminated and as a result these nodes keep their initial channel assignment f_1. Node e does not have any conflict neighbors and thus transmits using its initially assigned channel.

Chapter 4

Game Based Channel Allocation

A channel allocation approach that uses game theory to minimize a specific network interference metric will be presented in this chapter. More specifically, the state-of-the-art protocol GBCA [2] uses as interference metric the number of links that a receiver node can hear. In this chapter a modified version of GBCA under the SINR interference model will be introduced. Similarly to the implementation of the MinMax protocol, a priori scheduling knowledge is used so that interference links are determined in an effective way.

4.1 Problem Formulation

The *primal optimization problem* is to find a channel assignment f to minimize the total interference of a network. On the other hand, the *dual optimization problem* is to find a channel assignment f that maximizes the total removed interference. It is proved that both optimization problems are NP-hard [2]. Game theory is used to model the channel assignment problem as a repeated channel assignment game. The distributed GBCA algorithm solves the dual optimization problem in polynomial time but with a suboptimal result [2].

A receiver-based approach is adopted in this game and each receiver node is also a player of this game. The strategy of each player is its chosen channel. The game is evolved according to the Best Response (BR) dynamic: each player chooses a strategy (i.e. a channel) that maximizes its own payoff function given the other players' strategies.

The set of all interference links in an IC graph is denoted E_I (as explained in section 1.4). In every round of the game a subset of interfering links still remains in the network. For a given channel assignment f, E_{I_E} denotes the set of interference links eliminated in a round and thus cannot be further heard by their initial receiver (i.e. interference links that do not interfere with a transmission any more):

$$E_{I_E}(f) = \{e : ch(e) \neq f(r(e)), e \in E_I\}$$

where $r(e)$ is the receiver of link e, $ch(e)$ is the channel of the link. On the contrary, the set of interference links that still can be heard by their receivers (i.e. interference links that still exist in the IC graph) is denoted, E_{I_R} and defined as:

$$E_{I_R}(f) = \{e : ch(e) = f(r(e)), e \in E_I\}$$

Obviously $E_I = E_{I_E} + E_{I_R}$.

In order to minimize the total interference of the network, both the interference that player i suffers when a child is transmitting and the interference that a child of i causes to other players must be considered. Therefore, each player i has to choose a strategy (i.e. a channel), s, that maximizes the following payoff function, $u_i(s)$:

$$u_i(s) = -\sum_{e \in A(i,s)} I(e) - \sum_{e \in B(i,s)} I(e)$$
$$A(i,s) = \{e : e \in E_{I_R}(s), s(e) \in Child(i)\}$$
$$B(i,s) = \{e : e \in E_{I_R}(s), r(e) = i\}$$

where $Child(i)$ is the set of children of parent i,
$s(e)$ denotes the sender of link e,
$A(i,s)$ is the set of the outgoing interference links with i's children as senders,
$B(i,s)$ is the set of the incoming interference links of node i,
$I(e)$ is the actual number of transmissions, intended for $r(e)$, that the link e interferes with. In other words, $I(e)$ denotes the actual number of $r(e)$'s children whose transmissions are interfered by e.

Note here that in [2] it is assumed that an interference link e yields potential interference to the network which is always equal to the number of children of $r(e)$. That is because the interference metric adopted in [2] is the number of links heard by the receiver. However, this assumption does no hold in practice since a weak link heard at a receiver i may not be able to interfere with some transmission intended for i. This is exactly the point where our implementation is differentiated, since we use the SINR model described in section 2.2 to determine the interference links in a more realistic way.

As an example, the payoff values of each player in the network of Figure 2.8 can be shown in Figure 4.1. Initially, it can be observed that player c does not suffer any interference. However, its children cause interference to transmission links (f, s) and (d, f) respectively. Their parent c is "responsible" for these interferences (since children transmit to the parent's channel) and therefore decreases its payoff by one for every transmission link affected by a child. Node s suffers interference from links (a, s) and (d, s) and as a result $u_s(f_1) = -2$ (note that s's children c, f do not cause interference to any node). Player node f has both an incoming interference link from b (which interferes with transmission of node d) and an outgoing interference link from its child d to s (interfering c's transmissions) and as a result $u_f(f_1) = -2$ as well. If player i eliminates all of its interference links then the $u(i, s)$ value is equal to zero. Hence, for an interference-free channel assignmment every player in the game must have a zero payoff function.

Nash Equilibrium: NE is a stable state where for a set of strategies s^* and an arbitrary strategy s_i, the following inequality is always satisfied:

$$u_i(s_i, (s^*_{-i})) \leqslant u_i(s^*)$$

where s_{-i} denotes the strategies of all players except player i. Given an IC graph $G(V, E)$, the convergence of the Best Response dynamic to a Nash Equilibrium in at most $(|V| - 1)^2$ iterations is proved in [2].

A measurement of sub-optimality of a NE is given in [2] using the Price

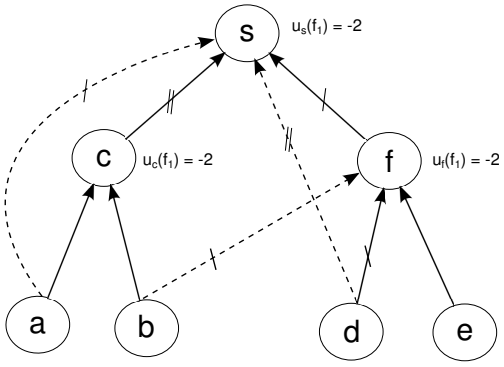

Figure 4.1: The IC graph of Figure 2.8 and payoff values for each player when the same strategy is used in the game based channel allocation approach.

of Anarchy [9]:

$$\frac{c-1}{c} \leqslant \frac{U(s^*)}{U(s^o)} \leqslant 1,$$

where c is the available number of channels,
$U(s^*)$ is the total removed interference for an arbitrary NE,
$U(s^o)$ is the total removed interference at optimal solution.

For example, assume that there are $c = 8$ available non-overlapping channels, at least $\frac{7}{8} * 100\% = 87.5\%$ of the intitial interference (i.e. interference under single channel) will be reduced. In other words, the NE is at most 12.5% worse than the optimal allocation.

4.2 The GBCA Algorithm

Two players are called *interfering players* if the transmission of a child of the first, interferes with a transmission of a child of the other. Since each player is a receiver node, a receiver-based conflict graph (as described in section 2.3) is used where each player communicates only with its interfering players (i.e. its neighbors in G_R) in order to implement BR. The phases that come before the game based channel allocation phase are the following:

1. Link-scheduling phase

2. IC graph construction phase
3. Receiver-based conflict graph creation

After the above steps, each player $i \in R$ has learnt both its interfering players and the actual links that determine its payoff. Notice that an *Outgoing_Link_Discovery packet* sent at the beginning of receiver-based conflict graph creation contains all the information needed. The basic idea of GBCA is that in each round, each player chooses a strategy based on the previous round strategies of its neighbors. The GBCA algorithm consists of the following phases:

1. Initialization

Assume $C = \{c_1, ..., c_m\}$ the set of available channels at each node. Initially all players choose the first channel as their strategy i.e. $s^0 = c_1$. Note that a common channel, c_0, is used for communication during the next phases.

2. RTC phase

At this phase each player knows the current strategies of its interfering players: each player may either have the previous round's strategy (or the initial strategy s^0 if it is the first round), or may have changed its strategy during the STC phase of the previous round. So every player node i has to decide the current round's strategy. For this reason, i calculates its current payoff value over all available channels considering the previous round's strategies and chooses the channel ch that results in the maximum payoff value.

If the chosen channel is different than i's previous round strategy, then node i must inform its neighbors in G_R that it wants to change its strategy. Thus, i broadcasts a *REQ* packet with its ID to its neighbors. On the contrary, if i chooses a channel that is the same with i's previous round strategy, then i keeps its previous round's strategy and does not broadcast any packet.

3. PTC phase

During this phase, each node i collects the REQ packets that received from its neighbors in G_R. Thus, the received *REQ* packets are stored in a list. Then node i has to reply to these messages. The idea is that each player node replies only to the sender of the REQ packet with the maximum ID among all received REQ packets. More specifically a unicast *PER* packet that contains i's ID is sent to the max ID node. In this way, the node with the maximum ID in every neighborhood of G_R is granted to change strategy. It must be noted that two neighbor nodes must not change strategy during the same round.[1] Thus, in this phase priority is given to the largest ID node.

4. STC phase

In STC phase, the players that were granted by all neighbors in G_R can now change their strategy. In other words if a node i received *PER* packets from all of its neighbors in PTC phase, then it is allowed to set the chosen channel, ch, of RTC phase as its current strategy, i.e $s_i = ch$. A broadcast *CHA* packet that includes the new strategy and i's ID must now be sent so that i's neighbors are informed.

5. RCC phase

This is the last phase in a GBCA round. Note that, every player keeps a table with the IDs and the strategies of its neighbors in G_R. Thus, if a player i receives a CHA packet during STC phase, then it updates the corersponding entry of that table.

The procedure is then repeated from the RTC phase and for $(|V|-1)^2$ iterations.[2] However, as it is shown in section 5.4, a smaller number of rounds is needed in practice. Furthermore, it has to be mentioned that two players may be 2 or more hops away. Then a indirect communicate scheme as the one described in section 2.3.1 must be employed: Every player transmits its packets in *max tx_power* so as reception through an interference link is guaranteed and then every node in the IC graph rebroadcasts the received packet.

[1] Remember that the same constraint is needed at MinMax algorithm too.
[2] Recall that V is the set of all nodes (including sink) in an IC graph

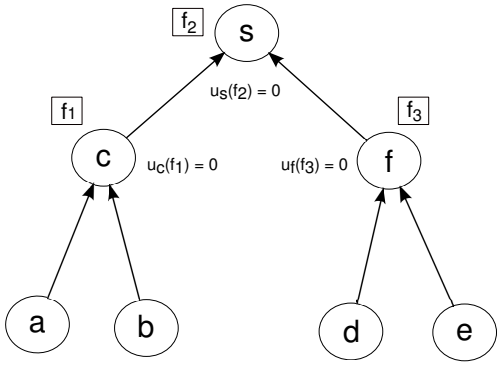

Figure 4.2: An interference-free receiver-based channel allocation for the IC graph of Figure 2.8 under the GBCA algorithm.

Note also that each player delays each transmission for a period related to its ID so that transmission conflicts are avoided.

Once the algorithm converges, each player i must inform its children in the IC graph for its chosen channel. For this reason a *Channel_Notification* packet is broadcasted in *normal tx_power* and includes i's final strategy as well as its ID. In this way every child node learns its parent's reception channel.

When GBCA is performed over small networks as the one of Figure 3.2, then a NE is always found. This means that an interference-free allocation as the one presented in Figure 4.2 is expected. Intially all players have the same strategy f_1. Then, in contrast to the MinMax algorithm, priority is given to the node with the largest ID and thus node s chooses the first available channel that is also different from its neighbors' in G_R. This is the channel c_2 with frequency f_2. In the next round, node f switches to the second available frequency, f_3 (since f_2 was occupied by s at a previous round). Finally, c does not change strategy, since its payoff at the initially assigned frequency f_1 has been maximized.

In can be observed that the payoff of every player has been maximized and each receiver node uses the frequency inside each rectangle to receive packets. Notice that in contrast to the MinMax allocation, three channels

are used.[3] This verifies the observation in [1] that link-based approaches provide better spatial channel reuse, since two child nodes are free to choose different channels and not exlusively their parent's channel.

[3]The MinMax allocation for the same network uses only two frequencies as has been shown in Figure 3.3.

Chapter 5

Simulation Model and Results

5.1 The Castalia Simulator

The evaluation of our distributed algorithms has been done in the Castalia simulator [12] which is based on the OMNET++ platform [13]. The basic modules of Castalia can be shown in Figure 5.1 [12]. There is a physical process which each node can sample in space and time and get various infos with its sensing devices. Note also that each node can communicate with each other only through the wireless channel module. This simulator has been chosen cause it provides *realistic* physical layer. Especially the radio model is based on real radios for low-power communication, such as the CC2420 [14], simulating realistic node behaviour.

The composite node module can be shown in Figure 5.2 [12]. It includes the radio, MAC and routing modules. There is also an application module and a sensor manager module which interacts with the physical process(es). The resource manager module is responsible for the node's power consumption and other node-specific quantities such as the clock drift [12], whereas the mobility manager determines the way nodes are placed or moved in space. Last but not least, the solid arrows in Figure 5.2 exist whether a module can communicate with another by exchanging messages, while the dashed ones represent simple function calling. The whole structure presented in Figures 5.1, 5.2 is implemented in Castalia with the use of OMNET++ NED language whilst each simple module (the application module for example) is defined in C++ code.

The distributed protocols examined in this thesis have been implemented at the application module of Castalia. A static routing (i.e each node forwards packets only to a specific node) has been also implemented at the

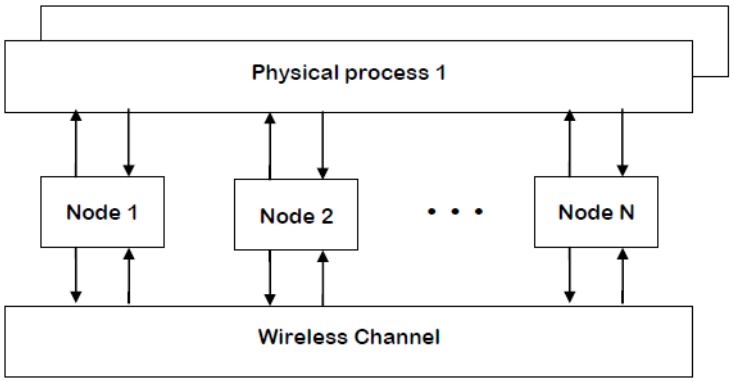

Figure 5.1: Castalia's basic module structure

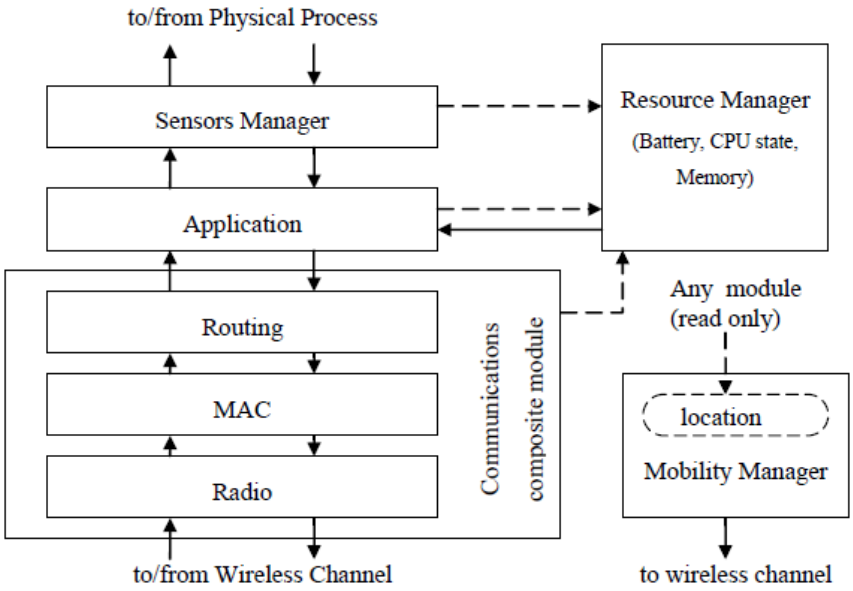

Figure 5.2: Castalia's node's module structure

routing module in order to create the tree-routing network model mentioned in Section 1.4. As far as MAC techniques are concerned, both contention-based and multiple access schemes were used:

- A simple CSMA was used for communication during all phases described at Chapters 2, 3, 4. The Tunable MAC module of Castalia which implements a CSMA mechanism was used as our contention based MAC. Note that no ACks and no retransmissions were employed.

- A TDMA-like scheme was implemented at the application module and consists of the link-scheduling protocol presented in Section 2.1. that performs time slot assignment among all nodes of the network in a distributed way. TDMA was used for relaying information at the sink after the frequency allocation phase.

At the radio module, a Castalia's parameter file which defines the real radio CC2420 by Texas Instruments [14] is used, with the values that are shown in Table 5.1.

The main characteristic of our implementation is that the already mentioned protocols were implemented in Castalia in a cross-layer way. For example, when a children discovery phase, which runs in the Application layer, finishes, the discovered children IDs must be sent to routing layer so that static routing tables are constructed. Moreover, the application layer channel allocation protocols need to poll the radio layer during IC graph construction phase in order to learn the heared links. Then, the radio layer sends back to application layer a message with the ID of such heared links. This can be done with the use of specific messages that allow communication between modules in a dynamic way (see the solid arrows in Figure 5.2). In addition to that, Castalia provides a set of commands that allow a node to adjust the MAC and radio parameters dynamically from an upper layer.

As far as the wireless channel model is concerned, Castalia employs the lognormal shadowing model as its radio propagation model [10]. According to this model, for any link (u, v), when u transmits with power t_u, in dBm,

Data rate	250 kbps
Modulation type	PSK
Bits per symbol	4
Carrier frequency	2.4 GHz
Bandwidth	20 MHz
Noise bandwidth	194 MHz
Noise floor	−100 dBm
Sensitivity	−95 dBm

Table 5.1: Radio parameter values in Castalia

the received power at node v is given by:

$$r_v = t_u - PL(d)$$

where $PL(d)$ is the path loss at distance d, measured in dB and is given by the following formula:

$$PL(d) = PL(d_0) + 10n\log_{10}(\frac{d}{d_0}) + X_\sigma \tag{5.1}$$

where d is the distance between the transmitter and the receiver node measured in meters, $PL(d_0)$ is the known path loss at the reference distance d_0 (also in dB), n is the path loss exponent (the attenuation rate of the signal) and X_σ is a zero-mean Gaussian random variable (in dB) with standard deviation σ. However, for simplicity in the evaluation of our algorithms we have useds $\sigma = 0$ and thus $X_\sigma = 0$. Then, we have the following path loss model:

$$PL(d) = PL(d_0) + 10n\log_{10}(\frac{d}{d_0}) \tag{5.2}$$

This more simplified model has been used for our simulations. As one can observe, no fading is used, i.e. for any link (u, v), the received power at v is always fixed for a specific distance of the nodes u, v and specific transmission power of u. The wireless channel's model characheristics can be shown in Table 5.2. All the values (except from σ) are the Castalia's default ones.

n	2.4
σ	0
d_0	1 m
$PL(d_0)$	55 dBm

Table 5.2: Wireless channel parameter values in Castalia

5.2 Simulation Model

When a channel allocation scheme converges, and the network is notified (in the ways described in Chapters 3, 4) every node switches to a phase called *data transmission phase*. In this phase, each node uses the channel determined by the channel allocation algorithm for communication with its neighbors in the IC graph. Furthemore, the *normal tx_power* is employed. During this phase only leaf nodes generate packets in a constant bitrate. The non-leaf nodes relay the received packets to the sink. Each node maintains a buffer where packets are stored: leaf nodes store the generated packets while non-leaf nodes store the packets received from their children.

A TDMA scheme is employed in this phase: each node transmits one packet from its buffer at the assigned time slot. The slot duration is 10 ms. In this amount of time, exactly one packet of 312 bytes can be transmitted. The tolal size of a packet is determined by the size of the data packet and the overhead added at each layer. The data packet size is 279 bytes as shown in Table 5.3. The overhead values for the application, network, MAC and radio layers that used in our implementation can be also shown in Table 5.3. As a result, the total overhead added at a data packet is equal to 33 bytes. Hence, the final size of a transmitted packet is determined as:

$$\text{total packet size} = \text{data packet size} + \text{total overhead}$$

and thus $\text{total packet size} = 312$ bytes in our case. The radio transmits packets at a data rate of 250 kbps. Therefore, a transmission will last for:

$$\text{transmission time} = \frac{\text{total packet size (bytes)} * 8 \quad \text{(bits)}}{\text{data rate} \quad \text{(bits/sec)}}$$

It is assumed that no other transmission delays exist and hence transmission time is equal with the chosen slot duration.

Our simulation results, can be divided into two categories: *algorithm evaluation* and *network performance evaluation*. Algorithm evaluation consists of metrics that evaluate numerically specific characteristics of the implemented channel allocation algorithms. Such characteristics are:

- The number of *iterations* needed for an algorithm's convergence.
- The *residual interference* of the network.

In our distributed implementations, a channel allocation algorithm is executed locally in each node. Therefore, not all nodes terminate the algorithm at the same time.[1] The *iterations* metric is the largest round number observed among all nodes of a conflict graph. When there are enough available channels a channel allocation scheme eliminates all interferences in a network.[2] Otherwise, interference links remain in an IC graph. The amount of the remaining interference in a network is evaluated using the *residual interference* metric. This metric is defined as the remaining $C(u, f)$ value of a node when the MinMax allocation is terminated or the remaining $|u_i(s)|$ value of a player when GBCA converges. Both maximum and average values among all nodes are recorded.
The network performance of each algorithm has been evaluated with the following metrics:

- *Latency* at sink
- *Packet delivery ratio (PDR)*
- *Throughput*

The latency of a packet is counted as follows: each application packet has a timestamp that records the time that the packet is released at the source. When the packet is delivered to the sink, the latter subtracts the timestamp

[1] A node a may need more rounds to minimize its conflict value (or maximize its payoff) than a node b.

[2] Network's topology and nodes connectivity determine to a great extent the number of channels needed for an interference-free channel allocation.

value from the current simulation time and thus calculates the packet's delay. Both average and max latency values are recorded. Note that lost packets are not taken into consideration, since no retransmissions mechanism is employed.
The PDR value is determined as follows:

$$\text{PDR} = \frac{\text{number of packets received at sink}}{\text{total number of generated packets}}$$

The fraction's numerator is the total number of packets that the sink has received during the whole simulation time. The denominator consists of the total number of packets that have been transmitted by all leaf nodes of the network. It is assumed that every leaf node transmits all the generated packets that exist in its buffer.
The network's throughput is defined as the total number of bits that the sink has received at the specific simulation time. As a result:

$$\text{Throughput} = \frac{\text{packets received at sink} * \text{total packet size}\,(\text{bytes}) * 8 \quad (\text{bits})}{\text{simulation time} \quad (\text{sec})}$$

The allready discussed parameters are summarized in Table 5.3.

The 4-hop topology illustrated in Figure 5.3 was used in our experiments. There are 18 nodes and a sink. Node m has 3 children, which is also the maximum number of children in this topology. Every 200 msec a packet is generated at a leaf node. The simulation time denotes the duration of the *data transmission phase*. Typically, the CC2420 radio chip [14] provides 16 non-overlapping channels. However, in order to avoid adjacent channel interference [3], no two adjacent channels are used and thus the number of available channels is reduced to 8.

All simulations have been conducted for two SINR thresholds (10 and 15 dB) and a variable number of available channels (from 1 to 8). These SINR thresholds have been chosen in contrast to the threshold of 5 dB that is used in [1]. A receiver becomes more sensitive to interference when a SINR threshold is increased. That is, more potential interference links are determined as interference links with the use of equation 2.1. In other words,

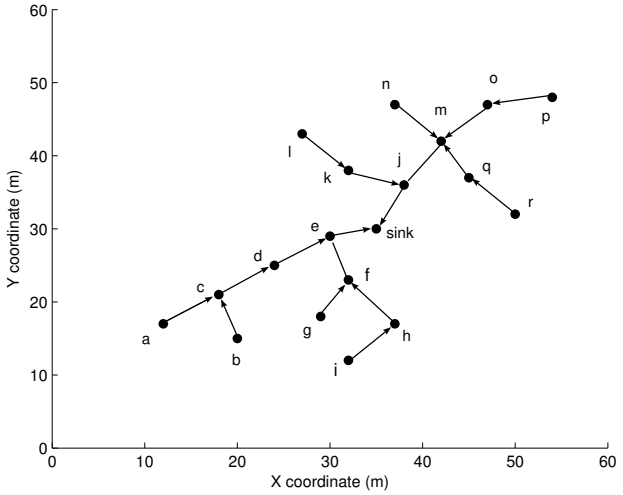

Figure 5.3: The 4-hop topology used in our experiments.

for the same topology, an IC graph created for $\theta = 15$ dB contains more interference links than the respective one for $\theta = 10$ dB.

Moreover, for the network performance evaluation results, the number of flows has been set to 6. The rate of each flow is 5 packets/sec. Note that each flow "starts" from a leaf node and thus for the topology in Figure 5.3, the maximum number of flows is 8. For this reason all possible combinations $\binom{8}{6}$ have been calculated and thus each simulation has been run under 28 different setups. Each result presented in next sections is the average value of these 28 simulations.

5.3 MinMax Evaluation

Figure 5.4 illustrates the number of rounds that MinMax algorithm needs to converge. The SINR threshold is denoted by θ. It can be seen that in all cases a fixed number of iterations is needed. In our implementation each node chooses a channel in a specific priority, which is determined by its ID (as described in section 3.2). That is, in each round a node does not negotiate channels with all of its neighbors but arbitrarily chooses the one

Data packet size	279 bytes
Application layer overhead	8 bytes
MAC layer overhead	9 bytes
Network layer overhead	10 bytes
Physical frame overhead	6 bytes
Slot duration	10 ms
Normal tx_power	−10 dBm
Max tx_power	0 dBm
Constant bitrate (CBR)	5 packets/sec
Number of nodes	19
Number of flows	6
Available channels	8
Simulation time	600 sec

Table 5.3: Simulation parameter values in Castalia

that minimizes its own conflict. Hence, the algorithm needs a fixed number of rounds in order to be terminated in every node. Besides, the iterations to converge metric depends on the convergence of the node with the largest ID (which is also fixed). For the specific topology, the algorithm is terminated after 14 rounds at node 18 and hence the algorithm's iterations to converge value is equal to 14.

The residual interference metrics can be shown in Figures 5.5, 5.6. When the MinMax algorithm converges, the residual interference for each node u is equal to its $C(u, f)$ value for the chosen channel f. The average conflict per transmission link and the maximum conflict among all transmission links are presented in Figure 5.5.[3] We can see that the MinMax channel allocation is interference-free when there are at least 5 available channels. Besides, the average values for $\theta = 15$ dB are always higher than the ones for $\theta = 10$ dB. That is because more interference links exist for higher SINR thresholds. However, the maximum value is the same for both thresholds. As one can see the MinMax protocol quickly minimizes the maximum interference values: when the number of available channels increases from 2 to 3 a maximum

[3]Since only sender nodes participate in the implemented MinMax channel allocation and each sender node has only one transmission link, the terms "per node" and "per link" have the same meaning.

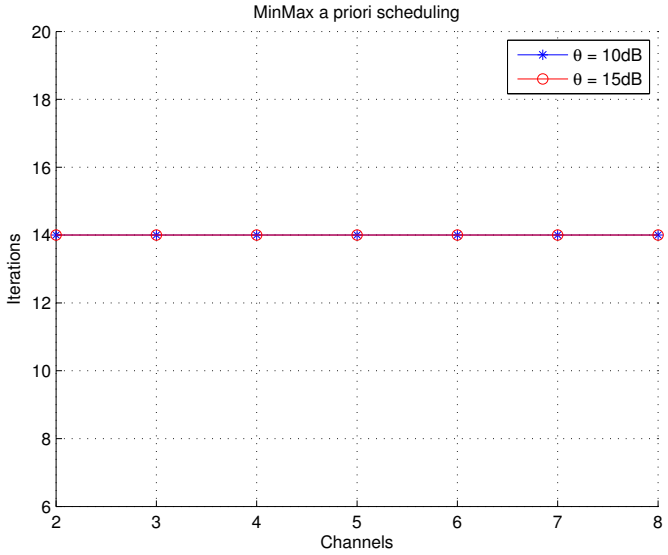

Figure 5.4: MinMax: Iterations to converge vs. Available channels

interfererence value decrease of 75% is observed. In figure 5.6 another interference metric is illustrated, called residual interference ratio and denoted as follows:

$$\text{residual interference ratio} = \frac{\sum_{i=1}^{N} C_{(i,f_i)}}{\sum_{i=1}^{N} C_{(i,f_1)}} \quad , \quad 1 \leqslant f_i \leqslant |F|$$

where N is the number of nodes in the G_L. The numerator of the fraction represents the total remaining conflict in a network when each node has chosen a channel f_i. The denominator denotes the total conflict metric for all nodes of the network under a single frequency. It can be seen that the presence of 2 channels results in an almost 55% decrease of the initial network's interference. The remaining interference is also reduced per about 68% when 3 channels are available. Note that at 4 available channels the network's remaining conflict is the same for both thresholds (as it can be seen at Figure 5.5), and thus the ratio value for $\theta = 10$ dB (in Figure 5.6) exceeds the one

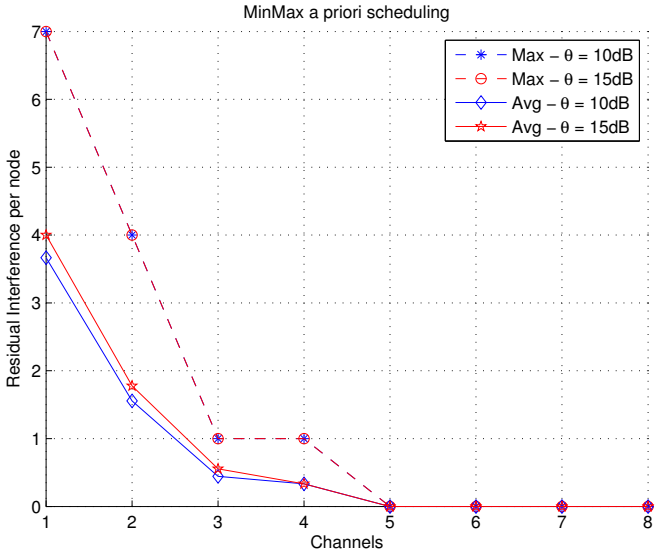

Figure 5.5: MinMax: Residual interference vs. Available channels

for $\theta = 15$ dB.[4]

The performance of the 4-hop network under MinMax channel allocation is now examined. Results for the packet delivery ratio and the throughput metric are presented in Figures 5.7 and 5.8. At first, it is observed that as the number of available channels increases, more packets are delivered to the sink. High packet delivery ratio and throughut values are also observed when interference is eliminated. Note that less than 20% of the created packets are delivered under a single frequency. This percentage could be higher if existing interfering links had been taken into account during the *link scheduling phase*. But this has not been done, in order to stress out the actual impact of a multifrequency assignment to the performance of a network.

Note also that with the strict SINR threshold of 15 dB a PDR of 100% is achieved.[5] When $\theta = 10$ dB, potential interference links that result to a SINR value slightly over 10 dB are assumed as non-interfering. However,

[4]Remember that a SINR threshold increase leads to larger $C(i, f_1)$ values.

[5]Remember again that network dynamics do not change during the simulation and transmission links are quite strong.

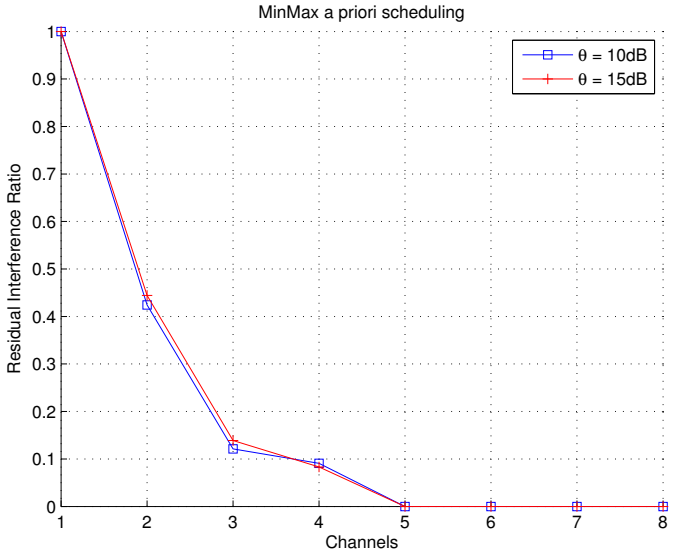

Figure 5.6: MinMax: Residual interference ratio vs. Available channels

Figures 5.7 and 5.8 show that such links may cause interference in practice. That is, a correct packet reception cannot be always guaranteed when the SINR value is close to 10 dB and thus a higher threshold is needed for an actual interference-free allocation.

Figure 5.9 depicts the network's performance in terms of packet latency. While the number of channels increases, more packets are delivered to the sink from more leaf-nodes and thus the average delay is increased. The max delay also increases in a similar way. Note that the lost packets are ignored, since no retransmissions mechanism is used. When the network becomes conflict-free, one can observe that the latency values have been stabilized. This is due to the fact that when more that 5 channels are available, the PDR and throughput values at the sink are fixed.

5.4 GBCA Evaluation

In this section the performance of the GBCA algorithm is also evaluated for the same 4-hop topology and simulation parameters with the MinMax.

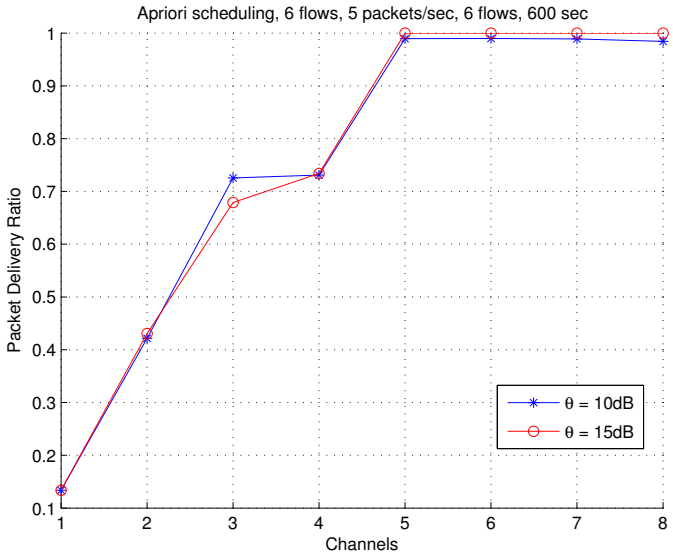

Figure 5.7: MinMax: Packet Delivery Ratio vs. Available channels

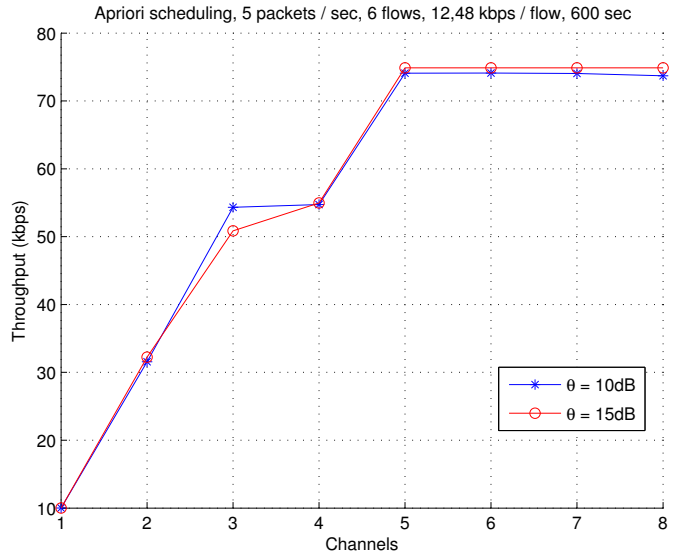

Figure 5.8: MinMax: Throughput vs. Available channels

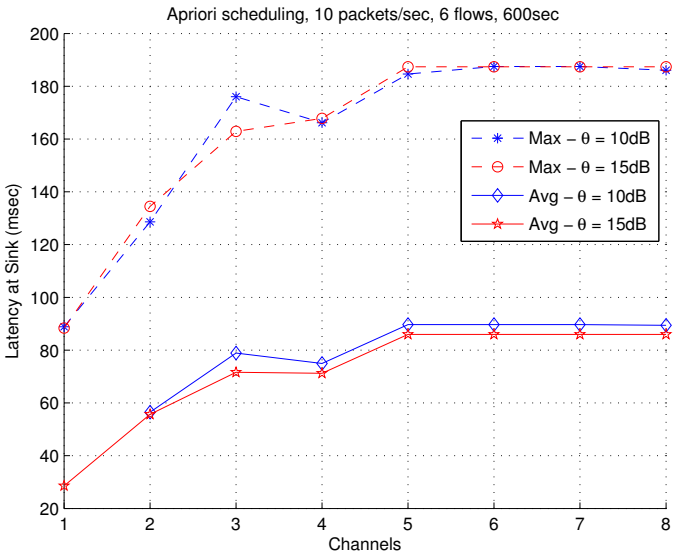

Figure 5.9: MinMax: Latency vs. Available channels

Figure 5.10 illustrates the actual number of iterations needed for convergence. Firstly, one can observe that GBCA converges very fast in comparison to the theoretical bound, $(|V|-1)^2$ which in our 4-hop topology is equal to 324. It can be noticed that the number of iterations slightly increases with the increase of available channels and is stabilized when more channels are available. Note that for $\theta = 15$dB excessive interference is suffered by certain nodes and this leads to the largest iterations number for the specific topology when there are only 2 available channels.

Once the GBCA algorithm converges, the residual interference metric for a player i is equal to its payoff value, $u_i(s^*)$, for a strategy s^* in the NE. The average remaining interference per every player node as well as the maximum value among all players are presented in Figure 5.11. We can see that with 5 available channels all the interferrence is eliminated and thus the specific NE is optimal, according to the *Corollary 3* of [2]. The average and max values for $\theta = 15$ dB are always higher than the ones for $\theta = 10$ dB. This means that for different θ's, some nodes suffer more interference (larger payoff values) than others. The fact that the initial interference value (interference under

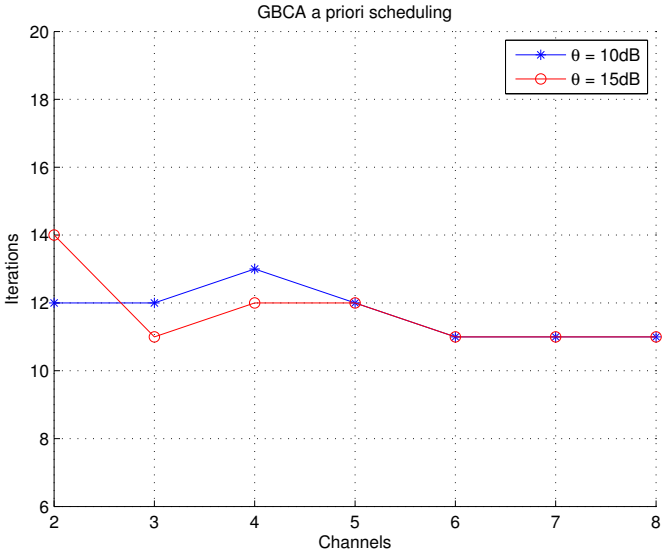

Figure 5.10: GBCA: Iterations to converge vs. Available channels

single channel) is the same for both thresholds has to do with the specific topology and the way interference links are calculated in payoff.[6]

The residual interference ratio is denoted as follows:

$$\text{residual interference ratio} = \frac{\sum_{i=1}^{\mathrm{N}} u_i(c_i)}{\sum_{i=1}^{N} u_i(c_1)} \quad , \quad 1 \leqslant c_i \leqslant C$$

where N is the number of nodes in the G_R. In accordance to the definion given for the MinMax protocol, the numerator of the fraction denotes the total residual interference in the network when each player has chosen the channel c_i. The total initial interference under single frequency is the fraction's denominator. The theoretical upper bound is $1/c$. It can be shown in Figure 5.12 that GBCA quickly eliminates the total interference of a network : with only three available channels, 5% of the intitial interference remains

[6]Remember that a link may not be able to interfere individually, but in composition with other links. In our implementation, the payoff function may include an interference link more that one times if this link belongs to more than one compositions.

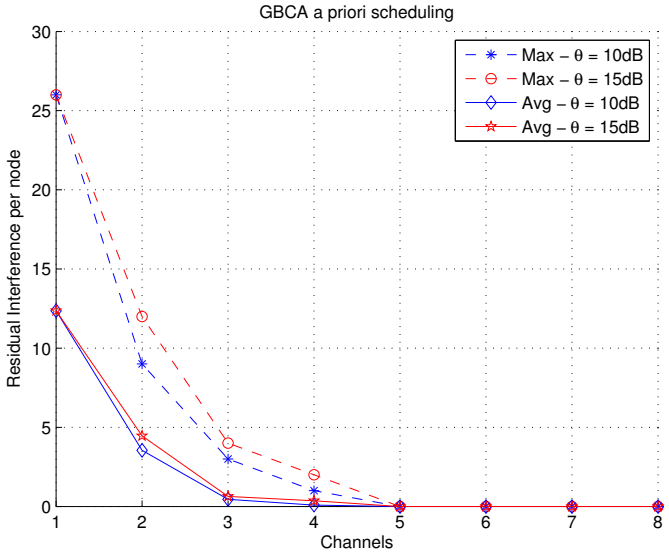

Figure 5.11: GBCA: Residual interference vs. Available channels

in the network, which is far less than the upper bound of 33%. Note that for both θ's, GBCA is always far from its upper bound and NE is closer to the optimal solution, which is achieved for 5 available channels.

The network performance metrics of PDR and throughput can be shown in Figures 5.13, 5.14. As the number of channels increases, more interference is eliminated and as a result more packets are delivered to the sink. For θ = 10dB a better channel allocation is achieved in terms of PDR and throughput when there are 3 or 4 available channels. On the contrary, when θ = 15dB, excessive interference is suffered by critical relay nodes of the network affecting the network's PDR and throughput to a great extent. However one can see that when there are more than 5 available channels, the channel assignment for θ = 15dB provides higher packet delivery rates.

In Figure 5.15 the game based channel assignment is evaluated in relation to the packet latency metric. Since no retransmissions mechanism is used, lost packets never reach the sink. One can observe that both latency values increase when the number of available channels increases, since more packets are delivered to the sink. It has to be noticed that both the average and

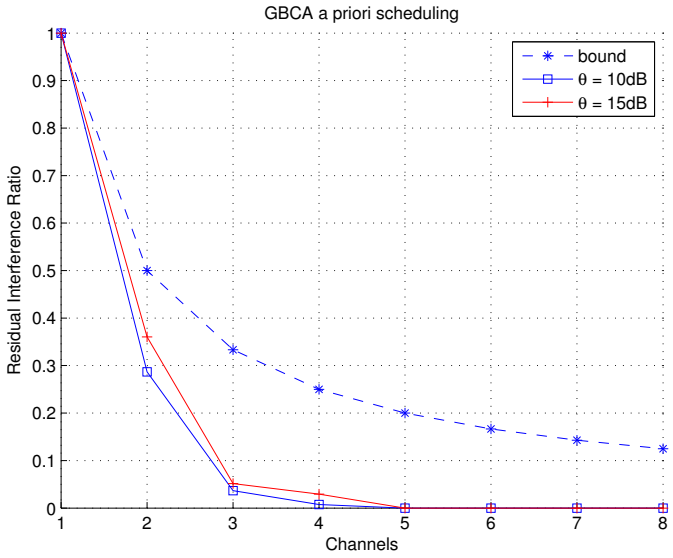

Figure 5.12: GBCA: Residual interference ratio vs. Available channels

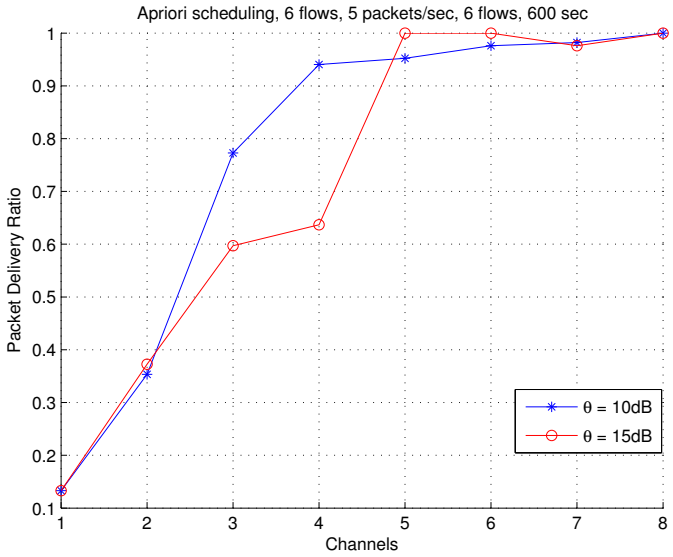

Figure 5.13: GBCA: Packet Delivery Ratio vs. Available channels

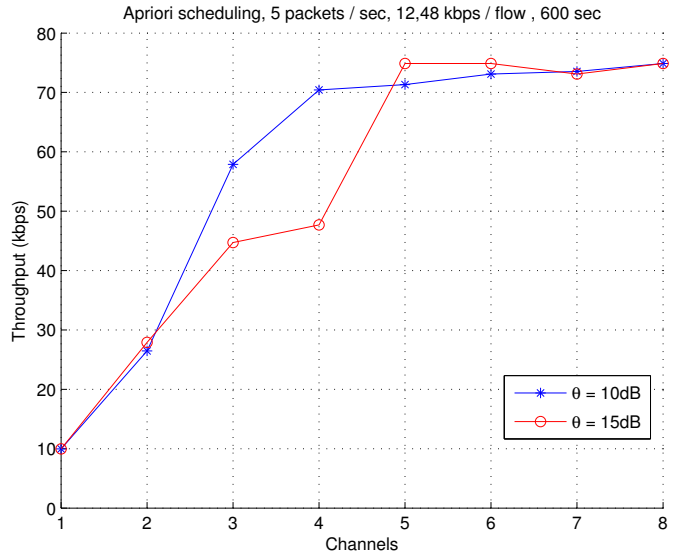

Figure 5.14: GBCA: Throughput vs. Available channels

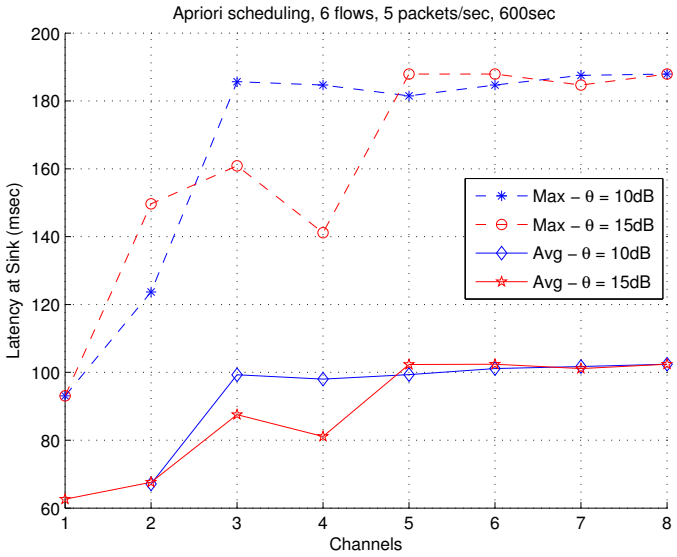

Figure 5.15: GBCA: Latency vs. Available channels

maximum latency values for θ = 15dB at 3 and 4 available channels are small, in agreement to the low PDR and throughput values of Figures 5.13, 5.14. Last but not least, the latency values are stabilized when high packet delivery ratios are achieved.

5.5 Comparison

In this section, a centralized greedy algorithm that minimizes the maximum interference, will be introduced. In addition to that, the centralized algorithm's network performance will be compared with the performance of the distributed MinMax and modified GBCA protocols.

Centralized Greedy Algorithm

A centralized greedy algorithm similar to the one mentioned in [1] was created in Matlab. It adopts a link-based approach and a link's conflict (i.e. a sender node's conflict) is defined in the same way as in MinMax algorithm (see Chapter 3). The algorithm works in the following way:

1. Intially, the IC graph of the network in Figure 5.3 is given as input to the algorithm along with a set of available channels. Every sender node is assigned the same channel at start and the conflict value of each transmission link is calculated.
2. In every round, every sender node detects the links that suffer the maximum conflict among all transmission links.
    - Every sender that affects a link with maximum conflict, checks if it can choose a channel such that switching to that channel the maximum conflict value is decreased. Note that the chosen channel must not increase any other link's conflict beyond the current maximum. If the maximum value cannot be decreased then the previous round's channel is kept.

- The senders that do not affect a link with maximum conflict, simply switch to a channel that results in the maximum decrease in their own conflict.

3. The algorithm converges, either when no sender can further decrease the maximum conflict or when all interference is eliminated.

Figures 5.16, 5.17 illustrate the number of iterations needed for convergence for all three protocols. We can observe that the centralized greedy converges very fast when a small number of channels is available. However, as the number of channels grows up, the greedy algorithm needs almost the same rounds with modified GBCA. Furthemore, the MinMax protocol needs 4 more rounds than the centralized one. This means that our distributed protocols that use local information converge quite fast in comparison with the centralized protocol that uses global knowledge of the network.

The centralized Greedy is evaluated at this point using the already known network performance metrics. In Figures 5.18, 5.20 a comparison of the three protocols in terms of PDR and throughput for $\theta = 10$dB is illustrated. The greedy channel allocation scheme results in the highest PDR and throughput values when there are at least 4 channels available. It can be also noticed that when there are not enough available channels (especially at 3 and 4 available channels case), modified GBCA has higher packet delivery values than MinMax. This is due to the fact that MinMax mainly aims at minimizing the maximum interference of a link : In the dense 4-hop topology there are many sender nodes with the same maximum conflict values and thus if there are not enough channels, not all of these values can be decreased. On the contrary, modified GBCA is more capable of reducing the total interference of the network with the existing channels and this yields to better performance values.

However, when a receiver becomes more sensitive to interference ($\theta = 15$ dB) it can be observed in Figures 5.19, 5.21 that the MinMax allocation is more efficient in terms of PDR and throughput than the modified GBCA. In this case where each transmission link suffers more interference, modified GBCA cannot reduce efficiently the total interference and hence a fair channel

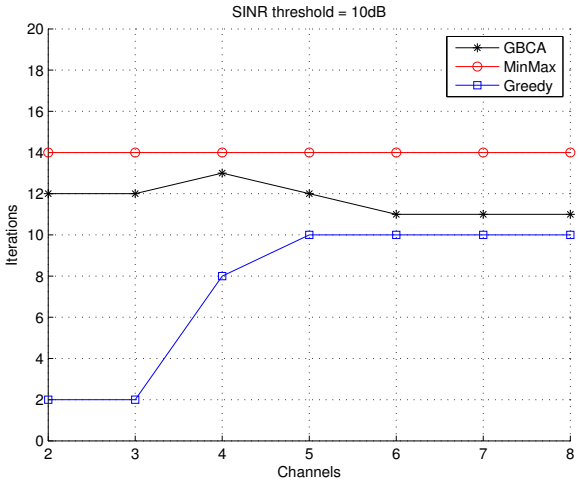

Figure 5.16: Comparison: Iterations to converge vs. Available channels, $\theta = 10$ dB.

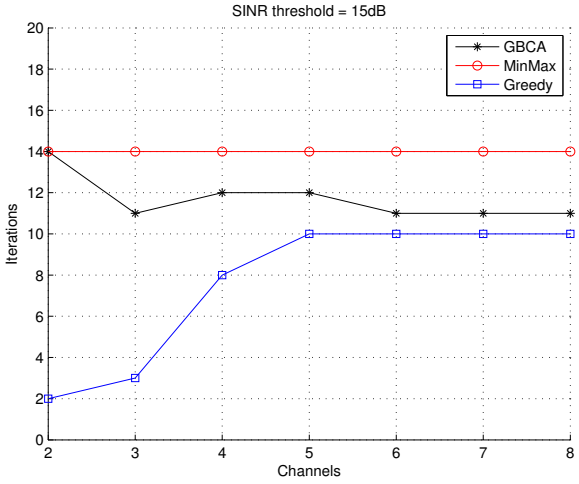

Figure 5.17: Comparison: Iterations to converge vs. Available channels, $\theta = 15$ dB.

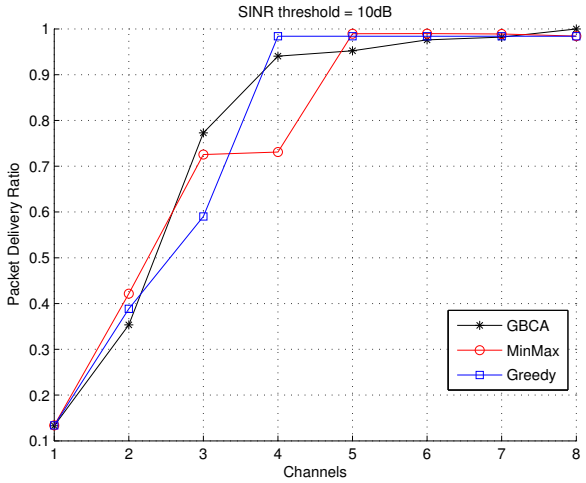

Figure 5.18: Comparison: Packet Delivery Ratio vs. Available channels, $\theta = 10$ dB.

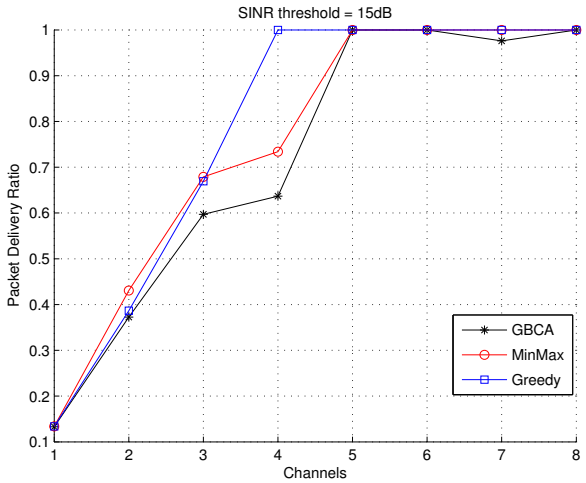

Figure 5.19: Comparison: Packet Delivery Ratio vs. Available channels, $\theta = 15$ dB.

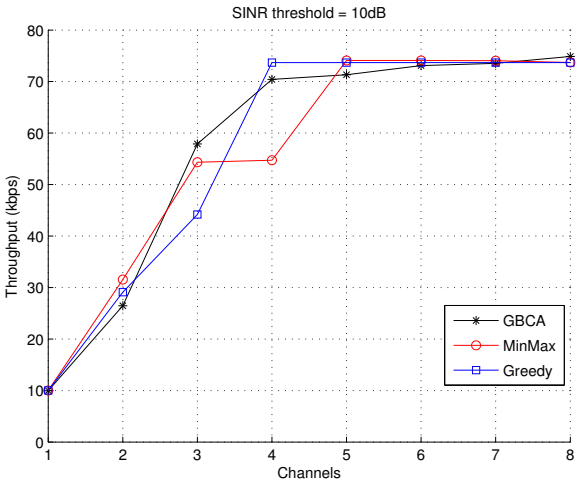

Figure 5.20: Comparison: Throughput vs. Available channels, $\theta = 10$ dB.

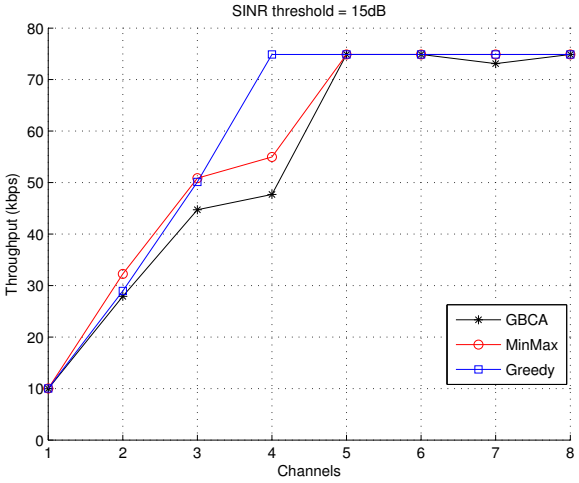

Figure 5.21: Comparison: Throughput vs. Available channels, $\theta = 15$ dB.

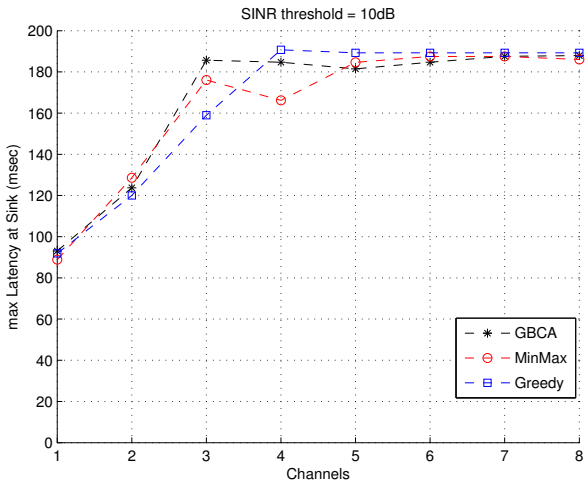

Figure 5.22: Comparison: Maximum Latency vs. Available channels, $\theta = 10$ dB.

allocation scheme that eliminates bottlenecks is preferable. This happens because modified GBCA takes into account both the interference suffered by a player and the interference caused by a player's child. This implies that very small payoff values are created and thus it is difficult to be maximized with a limited number of channels. As expected, the centralized greedy algorithm provides the most efficient channel assignment of all the three schemes using the least number of channels as well.

The maximum latency values can be shown in Figures 5.22, 5.23. Note that, for every available channel value, the largest among all max values belongs to the algorithm that yields the largest packet delivery ratio. Figure 5.24, 5.25 depicts the average packet delay at sink. It can be seen that all values are stabilized for more than 5 available channels, where high PDR and throughput values are observed. In addition to that, MinMax channel allocation gives the smallest average packet latency values for the specific topology.

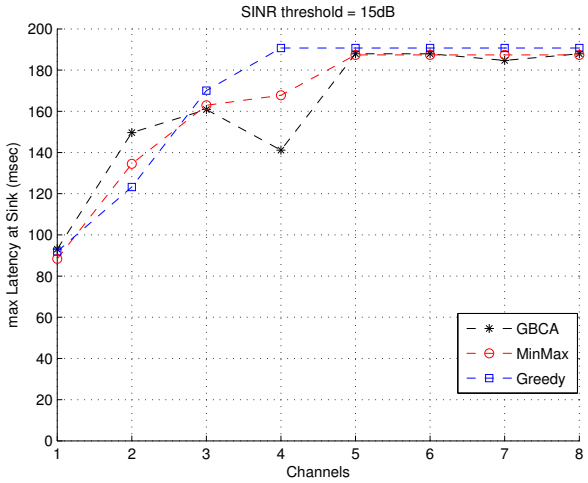

Figure 5.23: Comparison: Maximum Latency vs. Available channels, $\theta = 15$ dB.

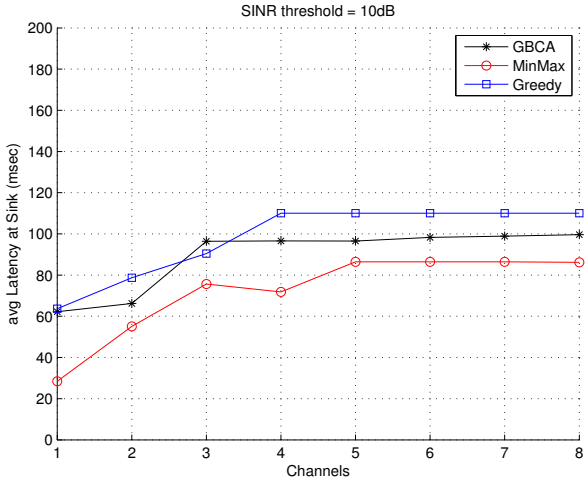

Figure 5.24: Comparison: Average Latency vs. Available channels, $\theta = 10$ dB.

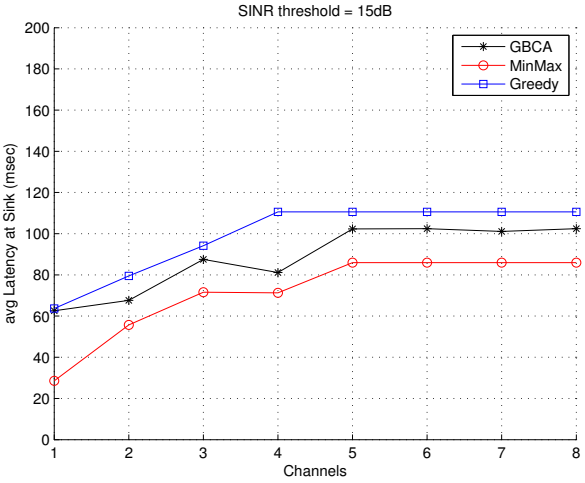

Figure 5.25: Comparison: Average Latency vs. Available channels, $\theta = 15$ dB.

Chapter 6

Conclusion and Future Work

In this work we have addressed the problem of distributed channel assignment in wireless sensor networks. We saw that this is a very challenging problem since there is a limited number of non-overlapping channels that can be used in practice. In addition to that, it has been shown that information about topology, routing and link-scheduling can heavily affect a channel allocation scheme. The problem has been proved to be NP-hard and therefore there is not any known algorithm that can solve it optimally in polynomial time. Different assumptions are made from the existing distributed algorithms in literature and sub-optimal solutions are offered. Many of these assumptions overlook network dymanics and the uncertainty factor of the wireless medium.

In Chapter 2, the creation of the IC graph in practice has been addressed. This graph actually depicts the pieces of information that have to be learnt and stored locally by every node. Therefore, realistic ways for distinguishing between transmission and interference links have been examined. Interference links are determined in a practical way with the use of the SINR-model. A priori scheduling information is also used in order to reduce the calculation overhead for all possible collision scenarios by ignoring potential interference links that will never cause interference at a specific slot. In this way channel allocation can be performed with a smaller number of channels. Last but not least, each node has to communicate with specific nodes at a channel allocation scheme. These nodes are extracted from the IC graph and a new graph called, conflict graph is created. In each channel allocation algorithm every node communicates only with its neighbors in the conflict graph. Packet receptions through interference links must be ensured in order to create such

graphs in practice.

In the two following Chapters the implementation of a distributed link-based channel allocation protocol, called MinMax and a receiver-based one called GBCA are described. These protocols are evaluated in terms of convergence and effective interference reduction as well as in terms of packet delivery ratio, network throughput and packet latency. We saw that both protocols converge quickly in comparison to a centralized protocol. Besides, from our experiments we have shown that in the case when sensitive to interference receivers are used, the fair MinMax channel allocation is preferable since it effectively eliminates excessive interference in the network and thus results in higher throughput values. With the presence of less sensitive to interference receivers, the GBCA protocol removes the total network's interference in a more effective way resulting in higher packet delivery ratios.

Two major requirements must be met so that such multi-channel allocation protocols are feasible in practical scenarios: 1) Consistent power level values for the potential interference links must be continuously kept so as actual interference links can be determined effectively. Thus, when the wireless channel's condition changes, both IC graph and conflict graphs must be reconstructed and thus considerable overhead is added to the network. 2) Communication between interference nodes must also be ensured. This may not be always possible in practice, since two neighbors in a conflict graph may be two or more hops away. The above two restrictions seem the most important limitations of the above algorithms.

Another disadvantage of the algorithms is that time synchronization is needed at all phases. Nodes must also be synchronized during the data transmission phase since a TDMA scheme is used. However, the receiver-based approach GBCA can be used with CSMA for transmitting data. In this case time synchronization is needed only for the negotiation period of the algorithm. The MinMax protocol can be used only along TDMA since each parent node has to know when to switch to a child's channel. Hence, in link-based approaches every parent node u must be able to operate on at most

$|Children(u)| + 1$ frequencies.[1] This poses a notable overhead at each node and from this point of view receiver-based schemes are preferred.

As a future work there are many fields that could be examined. A first task is to evaluate the implemented algorithms under more realistic shadowing channel models. Larger and denser topologies could be considered as a next task. In this case, the efficiency of a channel allocation assignment could be further investigated along with a reliable routing protocol (an implementation of the CTP protocol [11] already exists in Castalia simulator). Evaluation on other metrics such as the energy consumption per byte could also be done. For networks where the available channels do not suffice to remove all interferences, existing interference links could be taken into consideration during the link-scheduling phase and thus a time slot assignmnet could decrease their number. Then a channel allocation scheme should be employed to eliminate the remaining ones.

The resource allocation problem for WSNs is still an open topic. Existing works approach it with either a joint time slot and channel assignment or by separating it in two phases as it has been done in this work.

[1] $|Children(u)|$ denotes the number of children of u. Note also that u may use a different frequency for transmitting packets.

Bibliography

[1] A. Saifullah, Y. Xu, C. Lu, and Y. Chen, "Distributed Channel Allocation Algorithms for Wireless Sensor Networks," Tech. Rep., WUCSE-2011-62, Washington University in St Louis, 2011 http://www.cse.wustl.edu/šaifullaha/papers.html

[2] Q. Yu, J. Chen, Y. Fan, X. Shen, and Y. Sun, "Multi-channel assignment in wireless sensor networks: A game theoretic approach," *IEEE INFOCOM 2010*, San Diego, USA, March 2010.

[3] Y. Wu, J. Stankovic, T. He, and S. Lin, "Realistic and efficient multi-channel communications in wireless sensor networks," *IEEE INFOCOM 2008*, pages 1193-1201, Phoenix, Arizona, USA, April 2008.

[4] A. Ghosh, O. Durmaz Incel, V. Anil Kumar, and B. Krishnamachari, "Multi-channel scheduling for fast aggregated convergecast in wireless sensor networks," *IEEE MASS 2009*, Wuhan, China, September 2009.

[5] O. Chipara, C. Lu, and J. Stankovic, "Dynamic conflict-free query scheduling for wireless sensor networks," *IEEE ICNP 2006*, Santa Barbara, California, USA, November 2006.

[6] G. Zhou, T. He, J. A. Stankovic, and T. F. Abdelzaher, "RID: radio interference detection in wireless sensor networks," *IEEE INFOCOM 2005*, Miami, Florida, USA, March 2005.

[7] D. Gong, M. Zhao and Y. Yang, "Topology Control and Channel Assignment in Lossy Wireless Sensor Networks," *Teletraffic Congress (ITC), 2011 23rd International*, San Francisco, USA, September 2011.

[8] K. Srinivasan and P. Levis, "RSSI is under appreciated," *The Third IEEE Workshop on Embedded Networked Sensors (EmNets 2006)*, Harvard University, Cambridge, Massachusetts, May 2006.

[9] E. Koutsoupias and C. Papadimitriou, "Worst-case equilibria," *STACS 1999*, pages 404-413, University of Trier, Germany, March 1999.

[10] M. Zuniga, B. Krishnamachari,"Analyzing the Transitional Region in Low Power Wireless Links," *IEEE SECON*, pp. 517-526, Santa Clara, California, USA, October 2004.

[11] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and Philip Levis. "Collection Tree Protocol." In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys 2009)*, Berkeley, California, USA, November 2009.

[12] Athanassios Boulis et al, "Castalia: A simulator for Wireless Sensor Networks and Body Area Networks," User's Manual. http://castalia.npc.nicta.com.au/

[13] http://www.omnetpp.org/

[14] http://www.ti.com/product/cc2420