TKN Telecommunication Networks Group

Technical University Berlin

Telecommunication Networks Group

Double Hopping: A new Approach for Dynamic Frequency Hopping in Cognitive Radio Networks

Daniel Willkomm, Mathias Bohge, Dániel Hollós, and James Gross

{willkomm,bohge,hollos,gross}@tkn.tu-berlin.de

Berlin, January 2008

TKN Technical Report TKN-08-001

TKN Technical Reports Series

Editor: Prof. Dr.-Ing. Adam Wolisz

Abstract

One of the major challenges in designing cellular **C**ognitive **R**adio (CR) networks is the avoidance of **S**econdary **U**ser (SU) interference to so called **P**rimary **U**sers (PUs) operating in the licensed bands. Usually, SU operation has to be interrupted periodically in order to detect PU activity and avoid the respective frequencies. Recently, **D**ynamic **F**requency **H**opping (DFH) mechanisms have been suggested to enable reliable PU detection and continuous SU operation at the same time. Applying DFH in a multi-cell environment adds the challenge of mitigating **C**o-**C**hannel **I**nterference (CCI). In this context, graph coloring based optimization, which previously solely has been applied to non-hopping networks, is used to develop sophisticated centralized as well as distributed DFH methods. In this paper, we introduce a new DFH approach to allow reliable PU detection and continuous SU operation in cellular CR networks: **D**ouble **H**opping (DH). We present a centralized optimal, as well a distributed version of DH and evaluate their performance in terms of frequency usage. We show that the performance of the sub-optimal distributed version is only slightly worse than the optimal performance, and, thus, outperforms existing distributed approaches by far.

Contents

Chapter 1

Introduction

Cognitive **R**adio (CR) has become a popular and promising approach to overcome the artificial spectrum scarcity. The key idea of CR technologies is to allow the usage of temporarily unused licensed spectrum by so called **S**econdary **U**sers (SUs) under the constraint that the spectrum has to be vacated, as soon as the owner of the band – referred to as **P**rimary **U**ser (PU) – returns. To meet this constraint, the spectrum has to be sensed periodically – at least very $t_{\max}$ – in order to detect potentially appearing PUs. In order to perform reliable sensing on a frequency, data transmission has to be interrupted. Depending on the PU detection requirements and the sensitivity of the sensing antenna, the sensing process can require up to hundreds of milliseconds. Obviously, such interruptions in data transmission severely degrade the **Q**uality **o**f **S**ervice (QoS) – especially for real-time or streaming applications.

To avoid periodic interruptions of the payload communication, **D**ynamic **F**requency **H**opping (DFH) has been proposed for cellular CR networks [1, 2]. The basic idea of DFH is the following: A cell performs sensing on frequency Y **in parallel** to data transmission on frequency X. After $t_{\max}$, the cell hops to frequency Y and performs sensing on frequency X and so on. Obviously, an additional frequency per cell is needed to realize this approach. Having a whole network of mutually interfering cells, mitigating **C**o-**C**hannel **I**nterference (CCI) becomes crucial. The frequency usage of interfering cells has to be coordinated to avoid collisions. By using a smart distribution of frequencies among the cells in the network, the total number of additional frequencies in use can be minimized in order to maximize the number of supportable cells. Additionally, reducing the number of frequencies used for secondary communication also reduces the probability of interference with PUs. Whenever a PU appears on a frequency that is currently used for secondary communication, the secondary system temporarily produces potentially harmful interference to the licensed PU communication. Reducing the number of frequencies used for secondary communication, thus, also reduces the probability of interference with primary communications.

In non-hopping networks, frequency planning, which is based on solving the **F**requency **A**ssignment **P**roblem (FAP) [3], is an important method to control CCI in multi-cell communication systems. A typical FAP scenario consists of a set of cells. Each cell has a certain bandwidth requirement that needs to be complied by assigning a sufficient amount of frequencies to it. Interfering cells should not be assigned the same frequencies for operation. Solving the FAP results in a system-wide frequency setting that fulfills all bandwidth requirements at a minimum total amount of assigned frequencies. Mathematically, the FAP can be

expressed as a graph coloring problem: the nodes represent the cells, and the edges between the nodes represent their interference relationships. Each node is assigned one (or multiple) color(s) such that two connected nodes never own the same colors. The optimization goal is to minimize the total number of colors in use.

The above graph coloring problem is computationally complex. Mathematically speaking, it is similar to the *list coloring* problem, which is known to be *np-complete* [4]. Finding the system optimum for practically relevant systems requires prohibitively long computation times even with modern computers. Therefore two approximation approaches are usually applied:

- Suboptimal centralized algorithms which have a significantly reduced computational complexity while handling the full interference graph. For this approaches a single point of knowledge is assumed, which usually involves a significant signalling overhead.
- Decentralized approaches, in which each node selects its frequency based only on partial knowledge of the interference graph. This allows for parallelization of the computation and leads to the most significant reduction of the computational time.

So far, frequency planning has been studied for "frequency static" cells, i.e. cells which do not (or very rarely) change their operating frequency over time. In such cellular networks, both above mentioned approximation approaches achieve remarkably good results in the sense of minimizing the number of frequencies necessary for assuring a given level of traffic, as compared to the real optimum. It is intuitively clear that the FAP approaches a new level of complexity, if DFH is applied. Thus, the issue of reducing the computational complexity becomes critical – making decentralized approaches, as described above, especially attractive.

The question investigated in this report is how to minimize the number of frequencies required for DFH operation in a network of CR cells. The contributions of this report are:

- The presentation of a new concept for DFH called **D**ouble **H**opping (DH).
- A **L**inear **I**nteger **P**rogram (LIP) formulation for minimizing the number of frequencies required in DH mode. This optimum serves as a benchmark for the distributed algorithms.
- A distributed heuristic for minimizing the number of frequencies called **D**istributed **F**requency **A**llocation (DFA). DFA is based on the DH concept.
- A performance comparison of the DFA: (a) with the optimum, and (b) with the **D**istributed **H**opping **A**pproach (DHA), introduced in [5].
- A full protocol specification for both, the DFA and the DHA.

The remainder of this report is structured as follows. Chapter 2 presents related work on frequency hopping and the FAP in wireless networks. The general DH approach is introduced in Chapter 3 and the centralized and distributed realizations in Chapter 4. In Chapter 5, we present a performance evaluation and compare the different approaches. Finally, in Chapter 6, we conclude the report.

Chapter 2

Related Work

In this chapter, we first summarize related work that has been done in the area frequency hopping in general followed by CR specific frequency hopping related work. Afterwards, we point out related work that has been done in the area of of graph coloring for CR networks.

2.1 Frequency hopping

The idea of frequency hopping has gained attention in the context of GSM cellular systems, Bluetooth®, **W**ireless **L**ocal **A**rea **N**etwork (WLAN), and recently also in the CR community.

In the **G**lobal **S**ystem for **M**obile Communications (GSM) standard, frequency hopping is an optional mode to mitigate fast fading and co-channel interference. If frequency hopping is enabled, the tranmit frequency per terminal is changed once every **T**ime **D**ivision **M**ultiple **A**ccess (TDMA) frame (which has a duration of 4.17 ms) according to a prespecified hopping sequence. The impact of this hopping sequence (also referred to as **M**obile **A**llocation **L**ist (MAL)) design is studied in [6]. The authors propose a scheme which generates frequency lists assuming the knowledge about the frequency lists of neighboring, i.e. interfering, **B**ase **S**tations (BSs) such that interference between the neighboring (hopping) cells is within some specified constraint. Further work on the assignment of frequency lists in GSM systems can be found in [7]. The authors extend the above work by analyzing the effect of MAL coordination among several cells. The paper also discusses an algorithm to modify MALs. In contrast, [8] investigates *dynamic* frequency hopping in GSM and compares it to random hopping. The frequency hopping pattern of a mobile is adapted based on measurements made at the base station and the mobile. Recalculations are done after every TDMA frame, updating the hopping pattern of the mobiles. The paper studies several degrees of dynamic adaptations from only exchanging the worst channel to exchanging the whole hopping pattern.. However, the paper does not consider a jointly performed frequency list assignment over several cells.

For similar reasons (i.e. mitigating interference and fading), frequency hopping is applied in Bluetooth® systems. Hopping is performed every 625 μs. Each bluetooth cell chooses one out of several pre-specified pseudo-random hopping sequences. Recently, the Bluetooth Special Interest Group (SIG) adopted a non-cooperative **A**daptive **F**requency **H**opping (AFH) method for second generation Bluetooth devices to combat the so called frequency-static interference that originates e.g. from WLAN systems or microwave ovens [9, 10]. AFH enables

Bluetooth devices to adapt to the environment by identifying fixed sources of interference and excluding them from the frequency hopping list (categorization in "good" and "bad" channels). This process of re-mapping also involves reducing the number of channels to be used for hopping. Regulation authorities have specified a minimum number of channels to be used by a Bluetooth system. If the number of "good" channels is less than the specified minimum, additionally some "bad" channels have to be used. In [11], an interesting solution called Adaptive Frequency Rolling (AFR) to combat *self*-interference in Bluetooth systems, i.e. interference caused by overlapping Bluetooth cells, is proposed. The authors propose to divide the 79 channels into distinct "hopsets". A Bluetooth cell remaims in a hopset for a certain amount of time, after which it switches to the next hopset. Each cell, thus, is "rolling" through the whole spectrum. If a collision between two cells is detected (increased **P**acket **E**rror **R**ate (PER)), the cell performs a random jump. The authors extend their proposal to combat frequency-static interference by introducing AFR with probing, where – similar to AFH – certain frequencies can be excluded from the hopset.

In [12], Mishra et al. study the impact of hopping WLANs on the fairness provided within a larger network of uncoordinated WLAN cells. The authors propose to assign a hopping pattern to each WLAN cell such that all WLAN cells change their operation frequency after some transmission cycles. By this mechanism the performance degradation of interference is somewhat spread over all cells in the network over a longer time span, leading to an increased system-wide cell-level fairness. Note that in WLAN systems the number of available frequencies is limited such that several adjacent WLAN cells will be exposed to mutual interference.

2.2 Frequency hopping in CR networks

In contrast to the application of frequency hopping in non-CR related networks (e.g. GSM, WLAN, or Bluetooth®), where its main function is to mitigate fading effects and balance interference (as described above), the scope of applying frequency hopping in CR networks is to allow continuous data transmission and at the same time assuring the unimpaired operation of the PU. To the best of our knowledge, so far, among existing frequency hopping applications solely Bluetooth®'s AFR approach [11] is somewhat related and could be modified for CR operation. AFR tries to *avoid* frequencies occupied by WLAN systems (which could be seen as PUs) and at the same time also avoids CCI between different Bluetooth systems. However, since AFR has been developed under non-CR related assumptions, a high number of modifications would be necessary to support CR operation.

Frequency hopping in CR networks has been first considered within the IEEE 802.22 standardization process [13]. IEEE 802.22 is an emerging standard for **W**ireless **R**egional **A**rea **N**etworks (WRANs) operating on a license-exempt and non-interference basis in the spectrum allocated to TV broadcast services (between $47 - 910\,\text{MHz}$). It aims in providing alternative broadband wireless Internet access in rural areas without impacting the existing TV (i.e. PU) services.

Based on that standard, we have been the first ones to introduce the general concept of **D**ynamic **F**requency **H**opping (DFH) in 802.22 in [1, 2]. In these papers, we present phase-shifted operation for interference-free sensing and collision-free hopping in combination with

a cooperative hopping approach for neighboring cells referred to as **R**evolver **H**opping (RH). Additionally, the concept of **D**ynamic **F**requency **H**opping **C**ommunity (DFHC) is proposed as an effective way to organize the hopping of neighboring cells. Based on the RH approach presented in [2], the **D**istributed **H**opping **A**pproach (DHA) is proposed in [5]. DHA is a distributed algorithm realizing RH in a cluster of IEEE 802.22 cells without the need of a central controller managing the hopping patterns of the individual cells. However, it is also shown in [5] that there is a large difference in the performance of DHA and the theoretical optimal application of RH. This circumstance was one of the main motivations for us to search for a better performing distributed solution for solving the frequency assignment problem, which is done in this paper.

2.3 Graph coloring

The utilization of graph coloring in order to find optimal frequency assignments in "frequency-static" networks is well documented in literature. Basic studies on its use in connection with cellular networks are reported in [14, 15]. For more in-depth studies, an excellent web page on the topic is maintained by Eisenblätter and Koster [3].

In [16] and [17] two different approaches are made to use graph coloring in CR networks. However, note that in both papers, "frequency-static" networks are considered. In [16], in contrast to our approach, each node in the graph repersents one CR terminal. Each terminal is subject to an individual PU interference. Depending on this interference, the frequencies owned by a PU are more or less valuable for a certain terminal. When assigning the frequency to a terminal, this relationship is recognized by different so called "rewards" that are credited to the terminal at the time of the assignment. In addition, power control is used to control interference. The objective in [16] is the maximization of the network utility subject to reward and fairness constraints for a given number of terminals and available frequencies. Recall that, in contrast to that, our goal is to mininmize the number of necessary frequencies in order to maximize the number of supported terminals. In [17], the maximization of the network utility is considered as well, but, in contrast, all terminals experience the same interference from PUs. In conformity with our approach, the total number of channels used in the network is minimized. However, since there is a differing graph coloring - network relation, and since the different channels can have different bandwidths, and the number of terminals is fixed, a completely different optimization problem is formulated.

Chapter 3

Double Hopping

In this chapter we describe the **D**ouble **H**opping (DH) approach for DFH in CR networks. After presenting the assumed system model, we introduce the general concept of DH. Afterwards, we provide a brief comparison with the RH approach described in [1, 2, 5].

3.1 System model

We consider a system model based on the IEEE 802.22 standard [18]. IEEE 802.22 networks are organized in cells of several kilometers in diameter. Each cell consists of one BS and a number of associated terminals. The BS possesses total control over all terminals in its cell. Communication is performed on TV channels that momentarily are not in use by the TV broadcasting **P**rimary **U**ser (PU). We refer to the selected channels as *working frequencies*. To select one such working frequency for communication, the BS forces the terminals to sense the spectrum, collects the sensing results, and decides on the working frequency to be used by the cell. In order to avoid harmful interference, the working frequency has to be periodically sensed for (re)appearing PUs.

The maximal interference time ($t_{\max}$) is the maximum time span an SU is allowed to interfere with a PU. The data transmission time (t_{data}) is defined to be the maximum period of time a CR cell consecutively uses a working frequency for data transmission, with ($t_{\text{data}} \leq t_{\max}$). The sensing time ($t_{\text{sens}}$) is defined as the minimum amount of time required to (1) perform sensing on a set of n_s frequencies[1] (by the BS and the terminals), (2) gather the sensing information at the BS, and (3) inform all terminals about the working frequency to be used for the next data transmission period. The quiet time (t_{quiet}) is defined to be the amount of time a frequency is not used for data communication in order to perform sensing, which obviously implies $t_{\text{quiet}} \geq t_{\text{sens}}$. In our investigation, t_{quiet} is the smallest time unit we consider, i.e. time is discrete and quantized into units of t_{quiet}. Hence, the data transmission time (t_{data}) is a multiple integer of t_{quiet}. N_q is defined as the number of quiet times per data transmission time ($N_q = t_{\text{data}}/t_{\text{quiet}}$). The frame time ($t_{\text{frame}}$) defines the periodic operation

[1]The time needed for sensing depends on the chosen sensing technique (which is beyond the scope of this paper). Additionally, it also depends on the number of frequencies to sense. For simplicity, we assume that all frequencies can be sensed in parallel. In real-world systems, the BS has to select a subset n_s of the frequencies to sense within the specified time interval.

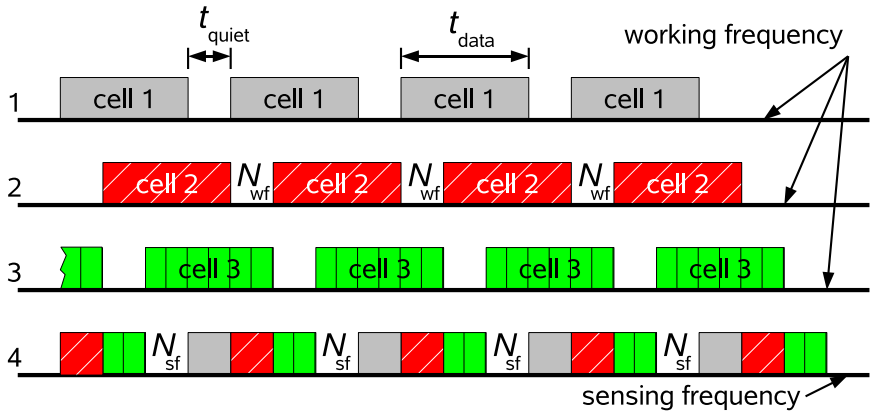


Figure 3.1: **C**ommon **D**ouble **H**opping operation

cycles and computes to $t_{\text{frame}} = t_{\text{data}} + t_{\text{quiet}}$.

In order to perform reliable sensing, the frequency being sensed, as well as up to k neighboring frequencies at both sides of the sensed spectrum must not be used for data transmission. Note that - for the sake of simplicity - we assume $k = 0$ for our investigations. Moreover, we do not investigate specific sensing mechanisms. Consequently, the reliability of sensing is not taken into account in our investigations.

3.2 Double Hopping operation

Double **H**opping (DH) is a new approach for cooperative DFH in CR networks. DH relies on the phase shifted operation of neighboring cells as introduced in [2]. Each cell has a dedicated *working frequency*, which is only used by that cell. Additionally, there is one *sensing frequency* for all cells. Once the data transmission time (t_{data}) of a cell expires, the cell hops to the sensing frequency, in order to perform sensing on its working frequency. It continues its current communication on the sensing frequency for t_{quiet}, before hopping back to its working frequency. Due to the time shifted operation all cells can consecutively switch to the sensing frequency in order to perform sensing on their working frequency. After one cycle, the sensing frequency has to be sensed simultaneously by **all** cells, i.e. all cells must share a common slot for sensing the sensing frequency. Consequently, each cell needs two sensing slots, one for the working frequency (N_{wf}) and one for the sensing frequency (N_{sf}).

Figure 3.1 shows an example for DH of 3 neighboring cells (i.e. all cells are mutually interfering). The operation period of all cells is shifted by multiple quiet times t_{quiet} against each other. During the sensing slot for the working frequency (N_{wf}) a cell hops to the sensing frequency (frequency 4 in Figure 3.1) to continue data transmission and perform sensing on its working frequency. In Figure 3.1, the sensing slots for cell two are marked. Note however, that the sensing slots for the sensing frequency (N_{sf}) are the same for all other cells.

Due to the phase-shifted operation a maximum number of N_q neighboring cells can be supported in a DH network. In order to be able to support N_q neighboring cells, all cells have to sense the sensing frequency in the same time slot, as shown in Figure 3.1. However, it is

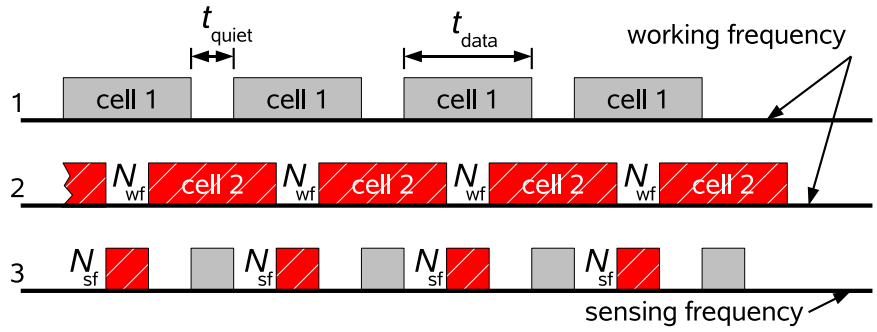


Figure 3.2: **I**ndividual **D**ouble **H**opping operation

also possible for each cell to have an *individual* time slot for sensing the sensing frequency, as shown in Figure 3.2. Again, the sensing slots for cell two are marked in the figure. Note that this time the sensing slot for the sensing frequency (N_{sf}) is exclusively used by cell two. In the figure, cell one would perform sensing on the sensing frequency immediately before using it for data transmission. We refer to the first DH variant as **C**ommon **D**ouble **H**opping (CDH) as all cells share a common slot for sensing the sensing frequency. To the second variant we refer to as **I**ndividual **D**ouble **H**opping (IDH), since each cell has an individual slot for sensing the sensing frequency.

Note that the presence of more than N_q neighboring cells would require some of them to hop at the same time, which is forbidden in phase-shifted operation mode. To increase the number of neighboring cells that can be supported in hopping operation, additional mechanisms to ensure a collision free operation would be needed. One possibility is to use distinct frequency sets, i.e. one group of neighboring cells using the frequency range between frequency a and b and another group of neighboring cells using the frequency range between frequency $b+1$ and c. Following this approach, up to a total number of $2N_q$ neighboring cells can be supported in the CDH variant. However, in the following we assume that a maximum number of N_q neighboring cells can be supported in hopping operation. If there are more than N_q neighboring cells, some of them have to operate in non-hopping mode, periodically interrupting data transmission to perform sensing. The frequency range for the non-hopping operation is distinct from the hopping frequency range.

3.3 Differences between Double Hopping and Revolver Hopping

The presented DH approach is closely related to the RH approach previously described in [1, 2, 5]. However, there are some significant differences between the two schemes: (1) in each frame time, each cell has to hop twice as much in the DH approach; (2) a cell uses each frequency for t_{data} before it hops to a new frequency in RH, whereas the sensing frequency is only used as long as the working frequency is sensed (for t_{quiet}) in DH; (3) the DH approach involves more (and more complex) coordination overhead: in addition to the working frequency (and N_{wf}), which has to be coordinated in both approaches, the sensing frequency (and N_{sf}) has

to be coordinated within each DH community; and (4) while the overall minimum frequency requirement is the same in both approaches, the frequency requirement **per cell** is much larger in the RH approach: each cell hops through the whole set of used frequencies. In the DH approach, in contrast, a CR cell only hops between 2 frequencies. This has several advantages:

- The sensitivity to PU interference is smaller: since each working frequency is exclusively used by one cell, solely this cell has to be shifted to another frequency in case a PU appears.
- Managing the coordination between different cells is easier: The only frequency that cells of one hopping community have in common is the sensing frequency (compared to the whole set of used frequencies in the RH approach).
- The possibility of frequency reuse is higher. In the RH approach, a cell cannot use the whole set of frequencies used by its neighboring cell belonging to a different hopping community. In the DH approach, only two frequencies are blocked: the working frequency and the sensing frequency.

Chapter 4

Hopping Pattern Generation

A major challenge to be solved operating a network of CR cells in DH mode is obviously the frequency selection of neighboring cells. As indicated above, the goal is to minimize the number of frequencies used in the whole network. One way to achieve this is graph coloring based optimization.

In this chapter – after introducing the simulation model – we present two realizations for DH based on graph coloring. The first one uses optimal frequency assignments calculated and distributed by a central controller. The second approach is based on distributed approximation algorithms.

4.1 Simulation model

The total number of CR cells in the investigated network is denoted by $|V|$; the total number of frequencies by F_{tot}, indexed from 1 to F_{tot}. Two CR cells are interfering if both operate on the same frequency and if at least one terminal (or the BS) of one cell is within the interference range of a terminal (or the BS) of the other cell. In the following, we regard a cell as the smallest entity and do not distinguish between the BS and terminals anymore. We assume that all information sent under the impact of interference is lost. Cells that are in interference range of each other are *neighboring cells*. We assume the interference from PUs to be global, i.e. an appearing PU is likewise present in all cells of the network. Furthermore, we assume that PUs do not change their frequency usage frequently over time[1]. This static PU model also serves the goal of analyzing the theoretical potential of the presented algorithms. Dynamic and only locally visible PUs are subject to future work.

We use interference topology graphs to model the interference relationships among cells. An example graph is shown in Figure 4.1. Here, each node represents a CR cell. The presence of an edge between two nodes (cells) indicates that they are within the interference range of each other. We define an *interference topology graph* $G = (V, E)$, where $V = \{v_1, ..., v_n\}$ denotes the CR cells and E is the set of interference relationships with $(i, j) \in E$ if v_i and v_j are in each others' interference range. Additionally, we define N_i to be the set of neighboring nodes of i, i.e. $j \in N_i$ if $(i, j) \in E$. Moreover, we assume the BSs of neighboring cells to have

[1]Recall that in IEEE 802.22 the most PUs are TV broadcasters which have a much larger interference range compared to 802.22 cells and do rarely change their usage.

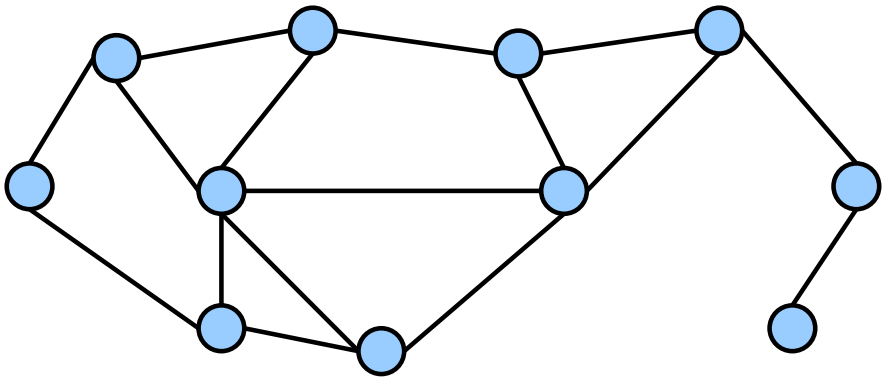


Figure 4.1: Example graph of CR cells

means to exchange control information. Note that we assume reliable control information related communication, i.e. we assume the lower layers to provide a reliable message delivery service to our control communication protocols. A CR cell can, thus, learn the interference relationships within its neighborhood by receiving their control messages.

4.2 Optimal Frequency Allocation

In this section, we introduce an optimization based centralized algorithm for the realization of the DH approach. We assume one central entity to be responsible for the hopping pattern generation for all cells in the network. The central entity has global knowledge about the interference graph and uses optimization tools to always assign optimal hopping patterns to all cells in the network. We refer to this approach as **O**ptimal **F**requency **A**llocation (OFA).

To determine the optimal frequency assignment for all cells in the network, the central entity has to solve the LIP given in Equations (4.1-4.9), where $x_{c,v}$ and $y_{c,v}$ are the binary assignment variables with

$$x_{c,v} \begin{cases} = 1 & \text{if node } v \text{ uses color } c \text{ as working color,} \\ = 0 & \text{if node } v \text{ does not use color } c \text{ as working color,} \end{cases}$$

and

$$y_{c,v} \begin{cases} = 1 & \text{if node } v \text{ uses color } c \text{ as sensing color,} \\ = 0 & \text{if node } v \text{ does not use color } c \text{ as sensing color.} \end{cases}$$

Assuming the interference graph G, solving the LIP assigns each node $v \in V$ a working and a sensing color $c_w, c_s \in C$.

$$\min \quad \sum_{\forall c \in C} b_c + \frac{a_c}{|C|} \tag{4.1}$$

$$\text{s. t.} \quad \sum_{\forall c \in C} x_{c,v} = 1 \qquad \forall\ v \in V \tag{4.2}$$

$$x_{c,v} + x_{c,w} \leq 1 \qquad \forall\ c \in C \wedge \forall (v,w) \in E \tag{4.3}$$

$$\sum_{\forall c \in C} y_{c,v} = 1 \qquad \forall\ v \in V \tag{4.4}$$

$$x_{c,v} + y_{c,v} \leq 1 \qquad \forall (c,v) \in C \times V \tag{4.5}$$

$$x_{c,v} + y_{c,w} \leq 1 \qquad \forall\ c \in C \wedge \forall (v,w) \in E \tag{4.6}$$

$$y_{c,v} + \sum_{\forall w \in N_v} y_{c,w} \leq N_q \qquad \forall (c,v) \in C \times V \tag{4.7}$$

$$a_c \geq x_{c,v} \qquad \forall (c,v) \in C \times V \tag{4.8}$$

$$b_c \geq y_{c,v} \qquad \forall (c,v) \in C \times V \tag{4.9}$$

Constraint (4.2) assures that each node is assigned exactly one working color, and constraint (4.3) assures that neighboring nodes do not get the same working color. Constraint (4.4) assures that each node is assigned exactly one sensing color, constraint (4.5) assures that the working and sensing color for a node differ, and constraint (4.6) assures that the sensing color differs from the working color of neighboring nodes. Constraint (4.7) ensures that no more than N_q neighboring nodes share the same sensing color. Finally, by the use of auxiliary variables a_c (constraint (4.8)) and b_c (constraint (4.9)) it is determined whether color c is used at all in the graph and are, thus, the variables to minimize. In order to not only minimize the total number of colors, but also the sensing colors used in the graph, we set the minimization of b_c as the primary objective.

Based on the solution of the LIP the central entity uses the color indices c in the graph coloring problem as frequency indices and sensing slots in the CR network. All cells v with $x_{1,v} = 1$ get assigned the frequency with index 1 and $N_{\text{wf}} = 1$; all cells with $x_{2,v} = 1$ get assigned the frequency with index 2 and $N_{\text{wf}} = 2$ and so on. Additionally, each cell v with $y_{i,v} = 1$ gets assigned the frequency with index i as sensing frequency, and the $N_{\text{sf}} = i$. The frequency assignments have to be distributed to all cells in the network. Additionally, the central entity has to recompute and redistribute the assignment as soon as there is any change in the interference relationship between the cells in the graph or a used frequency cannot be used anymore due to the appearance of a PU.

Note that solving the above LIP and, thus, the computation of the optimal frequency assignment might require extremely long computation times. Figure 4.2 shows the average run-times of an Intel Pentium 4 3.20 GHz that solves the above LIP as part of our simulations using the standard optimization software CPLEX [22]. According to this, the expected computation time increases exponentially with the number of nodes in the graph, which poses some scalability concerns. Moreover, the message overhead required per cell to collect the information about the whole interference graph at the central entity and to distribute the frequency assignments to the individual cells increases linearly with the average number of hops to reach the central entity. Hence, as a consequence of the long computation times

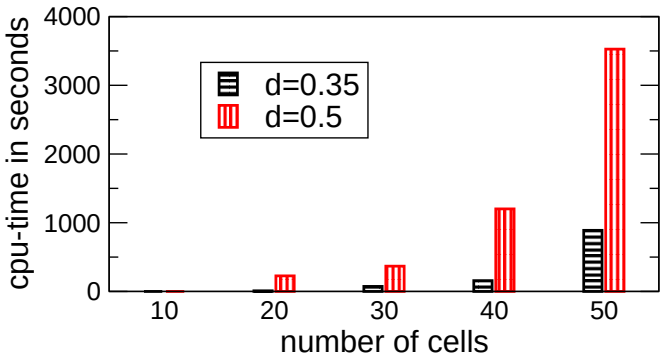


Figure 4.2: CPU times for solving the LIP using CPLEX on an Intel Pentium 4 3.20 GHz

and the high control message overhead in larger networks, we consider this approach mainly for comparison reasons (serving as a lower bound in frequency usage) rather than proposing it for practical usage.

4.3 Distributed Frequency Allocation

In this section a distributed algorithm realizing the DH approach is described. We assume that the cells only have information about their neighborhood, i.e. each cell knows its neighboring cells and has means to communicate with them. Additionally, a cell can learn the frequency usage of its neighbors by exchanging control messages. A detailed description of the protocol realization and implementation, including state diagrams, can be found in [19]. We refer to this approach as **D**istributed **F**requency **A**llocation (DFA).

DFA is based on the **D**istributed **L**argest **F**irst (DLF) algorithm [20] originally designed to solve static FAPs. This approach is known to perform near to optimal for static FAPs in practical problem instances. The basic idea of DLF is the following: After having discovered their cell neighbors, each node of the graph (i.e. each cell) collects information about the node degree (number of neighboring nodes) of its neighbors. Each cell then chooses its working frequency depending on its node degree, where the cell with the highest node degree selects its frequency first. In the case of equal node degrees, a random number is used for tie breaking. A cell always chooses the frequency with the lowest index available and distributes its choice within the neighborhood. This method ensures that two neighboring cells cannot select the same frequency (as only one frequency is chosen at a time).

The DLF is used for the initial working frequency (plus sensing slot – N_{wf}) selection. In addition, a sensing frequency (plus sensing slot – N_{sf}) has to be chosen. The first cell, thus – after choosing its working frequency – also "creates" a sensing frequency (again with the lowest frequency index available) and chooses a N_{sf} used to perform sensing on the sensing frequency. All subsequent cells choose the same sensing frequency and N_{sf} if applicable (in addition to a working frequency). If the sensing frequency or N_{sf} cannot be used the cell has to choose a different sensing frequency or N_{sf}.

Whenever a cell reaches its N_{wf}, it switches data transmission from its working frequency to its sensing frequency, and starts sensing. After t_{quiet} it switches back to its old working

frequency if no PU appeared on that frequency. Otherwise, it starts data transmission on a new working frequency (with the lowest frequency index available), or – if no new working frequency could be found – switches to the non-hopping mode.

Whenever a cell reaches its N_{sf}, it has to sense its sensing frequency. If it is not vacant anymore, but a new one was found, the cell has to broadcast a message indicating the new sensing frequency to its neighbors. In case no new sensing frequency could be found, the cell switches to the non-hopping mode.

In contrast to the optimal approach presented above, the computation time needed to generate a frequency assignment is distributed among all cells. The time needed at every cell is negligible and does not increase with the number of cells in the network. Furthermore, the control message overhead is also constant (one message per cell per frame time).

Chapter 5

Performance Analysis

In this chapter we present a performance analysis of the **D**istributed **F**requency **A**llocation (DFA) algorithm and protocol. We compare the DFA with the DHA and benchmark both against the optimal solution produced by the OFA.

5.1 Methodology

We have randomly generated interference topology graph instances using Culbersohn's graph generator [21] on a 1 by 1 unit plane, with the number of nodes varying between $|V| = 10$ and $|V| = 50$. The nodes are connected (i.e. the cells are interfering) if their euclidian distance is smaller than or equal to d, where we vary this distance between $d = 0.35$ and $d = 0.6$. We have generated 80 random graph topologies for each of those $(|V|, d)$ pairs. We assume that distinct frequency sets are used for the hopping and the non-hopping operation. Frequencies 1 to $F_{\text{nh}} - 1$ are used for the hopping mode; frequencies F_{nh} to F_{tot} are used for the non-hopping mode. This concept enables us to support up to $2N_q$ neighboring CR cells in time shifted operation, i.e. in a collision-free operation (assuming that the exchange of control messages is reliable). The simulation time for each simulation is set to 200 s. The quiet time is set to $t_{\text{quiet}} = 0.1\,\text{s}$. The frequency sets are determined by $F_{\text{tot}} = 50$, and $F_{\text{nh}} = 31$, resulting in 30 frequencies available for hopping operation and 20 frequencies available for non-hopping operation. The frame time is set to $t_{\text{frame}} = 2.0\,\text{s}$ which results in $t_{\text{data}} = 1.9\,\text{s}$ for the DFA and OFA and $t_{\text{data}} = 1.9\,\text{s}$ for the DHA approach.

We investigate two initialization methods. In the *optimal* initialization, all cells start operation synchronously using an initial hopping pattern determined by solving the LIP using CPLEX [22] or using the DLF, respectively. In the *random* initialization, a cell randomly starts operation within the first 10 s of the simulation. For OFA, the central entity has to recalculate all hopping patterns as soon as a new cell pops up. In DFA and DHA, after a cell pops up, it has to learn the interference relationships within its neighborhood by listening to control messages from neighboring cells. Based on this information it chooses the frequencies and sensing slots to use. The initialization phase of a cell is described in Appendix A.2 in detail.

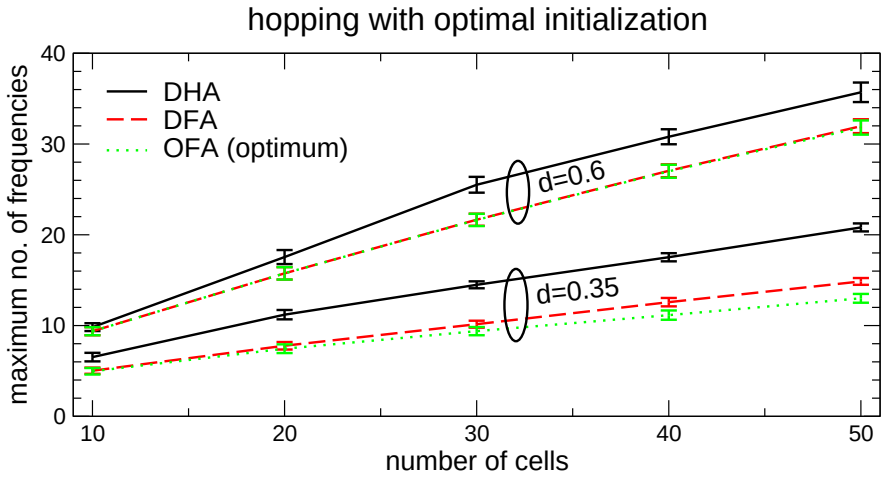


Figure 5.1: Maximum number of frequencies required, averaged over all instances for $d = 0.35$ and $d = 0.5$ for the hopping case.

5.2 Optimal initialization results

In this section we compare the number of frequencies used assuming a synchronous start of the cells and using a pre-calculated initial coloring. In Figure 5.1 we show the number of frequencies required for operating a *hopping* network in the centralized **O**ptimal **F**requency **A**llocation (OFA) or distributed (**D**istributed **H**opping **A**pproach (DHA) from [5] and our new **D**istributed **F**requency **A**llocation (DFA)) mode. The figure shows that the DFA approach achieves remarkably good results. For $d = 0.6$ there is almost no difference between the DFA and the optimal OFA. In the DHA case however, a lot more frequencies are required compared to the OFA results. This is mainly due to the fact that for the DFA approach the initial coloring computed by the DLF is kept as long as there are no changes in the interference environment (i.e. no PU or no additional CR shows up). As a consequence, the performance only depends on the initial coloring, which is nearly optimal for the investigated graph instances using the DLF algorithm. In the DHA approach this is not the case, as each cell *individually* chooses a new frequency before it has to perform a hop and thus – depending on the interference graph – two non-interfering cells originally operating on the same frequency might jump to different frequencies leading to a sub-optimal frequency usage.

5.3 Random initialization results

In this simulation scenario we investigate the number of frequencies used by the two distributed approaches assuming a sequential random initialization of cells. Figure 5.2 shows the number of frequencies required over the number of cells in the network. The bottom three graphs show the results for $d = 0.35$, the top three graphs for $d = 0.6$. The figure shows that the DFA implementation clearly outperforms the DHA also in the case where no initial coloring is used. Additionally, the results achieved by the DFA approach still can compete

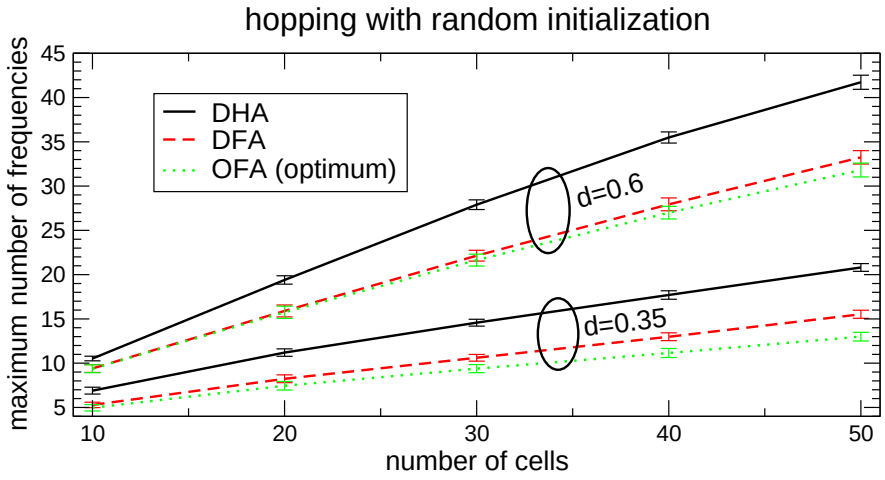


Figure 5.2: Maximum number of frequencies used for $d = 0.6$ (top) and $d = 0.35$ (bottom) averaged over all instances

with the optimal solution, especially for small $|V|$. The difference between the optimal solution and the DFA approach is smaller for $d = 0.6$, which is in accordance to the optimal initialization results.

Figure 5.3 shows the average percentage of non-hopping cells over the total number of cells in the network for $d = 0.6$. Until $|V| = 20$ there are no non-hopping cells, i.e. all cells operate in hopping mode. For bigger $|V|$ the number of non-hopping cells increases for all approaches (including the optimal OFA approach). The difference between the optimal and the two distributed approaches increases with increasing $|V|$. This is due to the fact that the sub-optimal frequency selection (i.e. sub-optimal N_{wf} selection) results in less cells within a neighborhood that can be supported in the hopping mode. The difference between the DHA and the DFA approach is due to the additional sensing slot needed for the sensing frequency (N_{sf}) in the DFA approach. Since we decreased the data transmission time for the DFA approach, there is one sensing slot less than in the DHA approach. However, for $|V| = 50$ the superior frequency selection of the DFA approach can be seen, as the difference between DFA and DHA approach decreases again.

In Figure 5.4(a) and 5.4(b) we compare the results achieved in case of DLF initialization and random initialization for the DFA and the DHA mode respectively. For the DFA approach we can see that the difference is marginal, i.e. even using a non-optimal initialization solely based on the (random) appearance of the cells achieves good results. For the DHA approach it is interesting to observe, that for $d = 0.35$ there is no difference between the DLF and the optimal initialization. For $d = 0.6$, however, the figure shows that the performance difference increases as the number of cells in the network increases.

Figure 5.5 compares the **P**robability **M**ass **F**unction (PMF) of the DHA and the DFA mode for different $|V|$ and different d. The figure shows that the DFA mode always outperforms the DHA mode. The performance difference between both approaches increases as $|V|$ as well as d increases. In Figure 5.5(a) it can be seen that the PMFs of both approaches still have a large overlap for both $= 0.35$ and $d = 0.6$. However, even here the probability for a

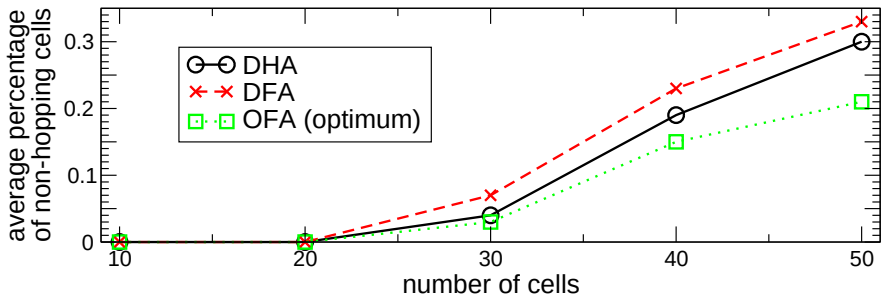
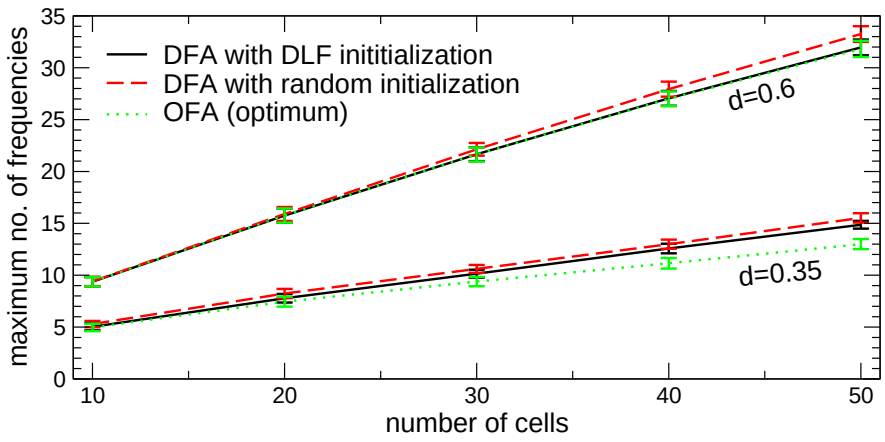
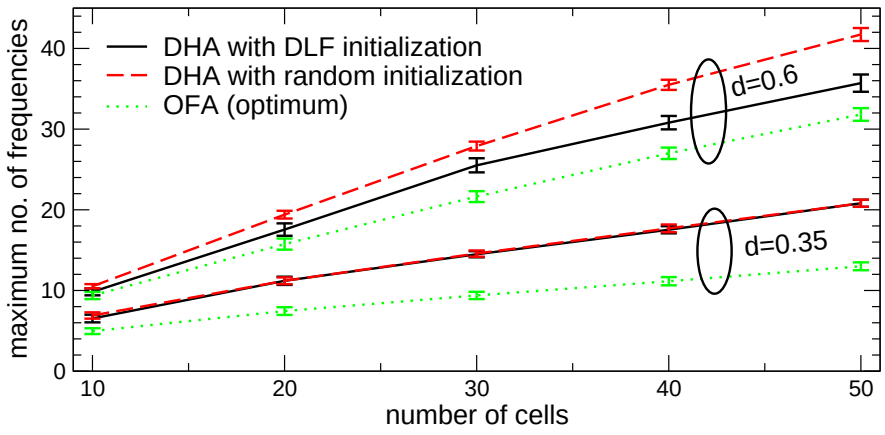


Figure 5.3: Average percentage of non-hopping cells for $d = 0.6$



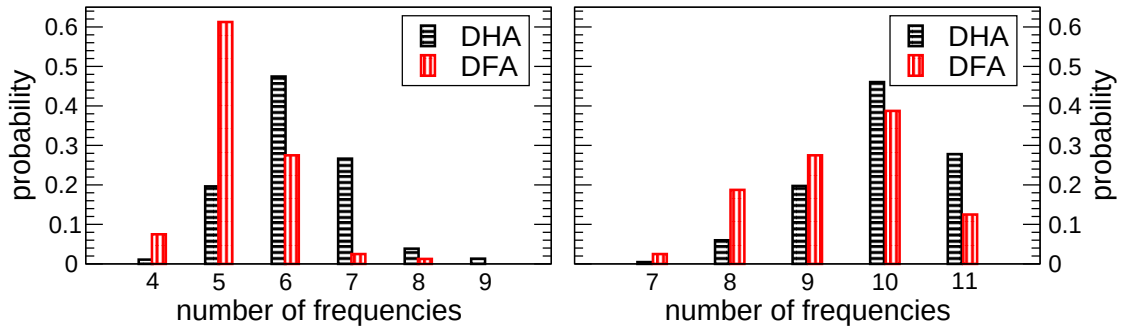
(a) **D**istributed **F**requency **A**llocation



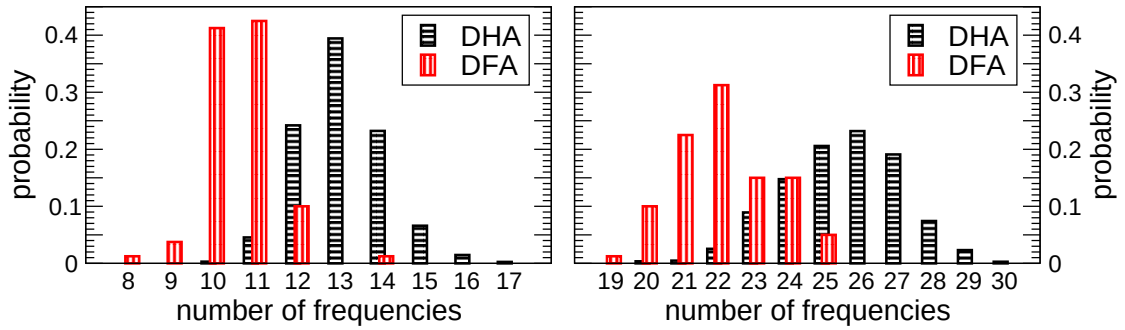
(b) **D**istributed **H**opping **A**pproach

Figure 5.4: Comparison of the maximum frequency requirement in case of DLF and sequential initialization for the DFA (top) and the DHA (bottom) approach.

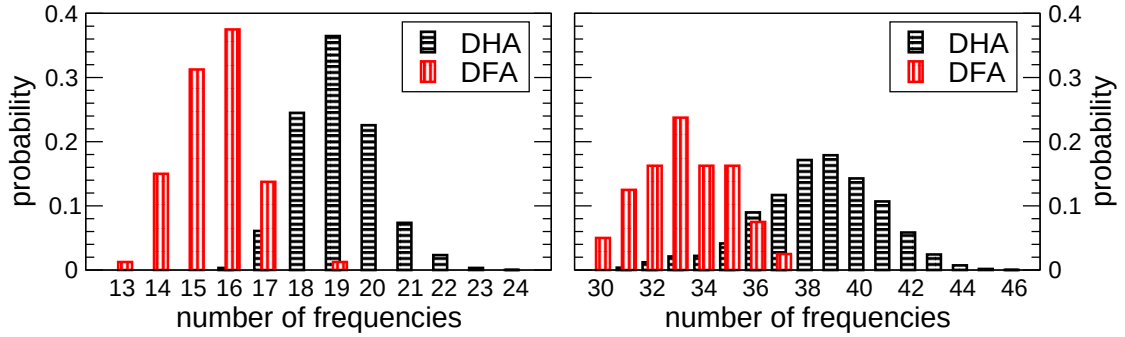
higher number of frequencies is lower for the DFA than for the DHA approach. Figure 5.5(b) shows the PMF for $|V| = 30$. Here, the overlap between the both approaches is already smaller than for $|V| = 10$. Finally, Figure 5.5(c) shows the PMF for $|V| = 50$. Here, the performance difference is biggest, i.e., the overlap between both approaches is smallest.



(a) left: $|V| = 10$, $d = 0.35$, right: $|V| = 10$, $d = 0.6$



(b) left: $|V| = 30$, $d = 0.35$, right: $|V| = 30$, $d = 0.6$



(c) left: $|V| = 50$, $d = 0.35$, right: $|V| = 50$, $d = 0.6$

Figure 5.5: PMF of the frequency requirement for $|V| = 10$ (top), $|V| = 30$ (middle), and $|V| = 50$ (bottom) nodes

Chapter 6

Conclusions

In this paper we have presented **D**ouble **H**opping (DH), a new approach to apply **D**ynamic **F**requency **H**opping (DFH) in CR cellular networks. The presented approach allows for continuous data transmission in CR networks while enabling reliable detection of **P**rimary **U**sers (PUs). Compared to **R**evolver **H**opping (RH), which was proposed for DFH in CR cellular networks earlier, DH has one major advantage: Although the minimum number of frequencies required to operate a network of cells in DH mode is the same as for the RH mode, the number of frequencies required **per cell** is much lower for the DH approach. In the DH approach each cell occupies two frequencies only, whereas in RH each cell hops through the whole set of cells used by the network. On the other hand, each cell needs to hop twice as often in the DH approach, and the coordination overhead to manage the frequency usage is larger than in RH.

We have applied graph coloring optimization to minimize the number of required frequencies in DFH cellular CR networks, in order to minimize potential **S**econdary **U**ser (SU) generated interference on PUs and to maximize the number of supportable cells. Based on that, we have presented two realizations of DH: centralized **O**ptimal **F**requency **A**llocation (OFA) and **D**istributed **F**requency **A**llocation (DFA). In our performance evaluation, we compare the DFA with the **D**istributed **H**opping **A**pproach (DHA) a distributed implementation of the RH approach. The OFA realization results are used as a lower bound to benchmark the two distributed realizations. Our performance evaluation results show that the proposed DFA can compete with the optimal results of the OFA and outperforms the DHA approach by far. The DHA approach needs significantly more frequencies, as it involves a higher dynamic in frequency usage (a single cell uses a plurality of frequencies). We, thus, have shown, that – as in the frequency-static case – distributed approaches for solving the **F**requency **A**ssignment **P**roblem (FAP) that achieve results comparable to the optimum exist.

Investigating the impact of PU dynamics is left as a future work issue. Another issue is to explore the impact of the amount of neighborhood information (i.e. the frequency usage of neighboring cells) on the approaches. Currently, each cell has knowledge about its one-hop neighborhood only. The related research question to answer is if the approaches significantly gain from e.g. having two-hop neighborhood knowledge. Additionally, our preliminary results motivate the introduction of cooperation between hopping cells by grouping cells into communities. Each community can be assumed to have regional information about its vicinity. The trade-off between the overhead to keep these informations up to date and the gain in performance is another interesting optimization problem.

Appendix A

Distributed DFH Protocol

In this chapter, we present a distributed protocol realization for DFH. The protocol supports DHA as well as DFA operation. Additionally, a non-hopping operation mode is implemented. If a cell cannot operate in DFH mode (either because no frequencies or no sensing slots are available) it tries to operate in non-hopping mode.

We assume that distinct frequency sets are used for the hopping and the non-hopping operation. Frequencies 1 to $F_{\mathrm{nh}} - 1$ are used for the hopping mode; frequencies F_{nh} to F_{tot} are used for the non-hopping mode. This concept enables us to support up to $2N_q$ neighboring CR cells in phase-shifted operation, i.e. in a collision-free operation (assuming that the exchange of control messages is reliable).

The described protocol has been implemented and tested using OMNeT++ [23] and the mobility framework [24]. In the state diagrams the following acronyms are used: *workingFreq* for working frequency (F_{w}), *sensSlotWork* for sensing slot for the working frequency (N_{wf}), *sensingFreq* for sensing frequency (F_{s}), and *sensSlotSens* for sensing slot for the sensing frequency (N_{sf}).

A.1 Message types and timer

A BS informs its neighbor BSs about its current frequency usage by broadcasting control messages on the control channel. The `neighborhood information` which is shared between neighboring cells is described in Table A.1. F_{s} and N_{sf} are only included if the system is operating in DH mode.

parameter	**description**
timestamp	a reference timestamp for the framing structure
state	the current state of the cell
F_{w}	the current working frequency
N_{wf}	the time slot for sensing the working frequency
F_{s}	the current sensing frequency
N_{sf}	the time slot for sensing the sensing frequency

Table A.1: Neighborhood information

name	description
`neighborInfo`	The *neighborhood discovery* message announces the existence and current frequency usage of a cell.
`jumpAnnounce`	The *jump announcement* message indicates that a cell jumped to a new frequency.
`newSensFreq`	The *new sensing frequency message* indicates that the cell changed its sensing frequency.
`endListen`	The `endListen` timer indicates the end of the initial listen period.
`endInit`	The `endInit` timer indicates the end of the initialization phase and the start of the hopping / non-hopping operation.
`sensWork`	The `sensWork` timer indicates the start for sensing the working frequency.
`sensSens`	The `sensSens` timer indicates the start for sensing the sensing frequency.

Table A.2: Messages and timers

We define three message types and four timers as described in Table A.2. All messages contain the *neighborhood information* as described in Table A.1. A `neighborInfo` message additionally contains the start time of the cell, and a random number (used for tie breaking).

The sensing slots are to be understood with respect to the timestamp, where the timestamp marks the beginning of a frame, i.e. the beginning of the first slot in the frame. The start time of sensing the working frequency (and thus also the time the `sensWork` timer is scheduled to) is thus "timestamp $+ (N_{\text{wf}} - 1) \cdot t_{\text{quiet}}$". The sensing is finished one slot (t_{quiet} time units). At that time also the jump to the new frequency is performed.

A.2 `initialization` state

After a cell is powered on, it is in the `initialization` state. It immediately broadcasts a *neighborhood discovery* (`neighborInfo`) message to announce its existence to its neighbors. Since no frequencies or sensing slots are selected yet, the respective values in the *neighborhood information* are set to -1. Subsequently, it sets the `endListen` timer and listens for messages from its neighbors to get a list of active neighbors and their frequency usage. If a message from a neighbor arrives, the frequency usage is updated, which means, that (1) the *neighborhood information* for the neighbor is updated and (2) the frequency list is updated. Updating the frequency list means, that the working frequency and N_{wf} of the neighbor is marked used and that the sensing frequency and N_{sf} are marked as sensing frequency and N_{sf}, so that they can also be used by this cell if applicable.

Once the initial listen period is over, the cell has full knowledge about its neighborhood and, thus, also about the frequencies used within its neighborhood. It then tries to find an available hopping working frequency and proceeds as described in Chapter A.2. If no hopping frequency is available as working frequency, it tries to find a non-hopping working frequency and proceeds as described in Chapter A.2. A state diagram for the initialization state can be found in Figure A.1.

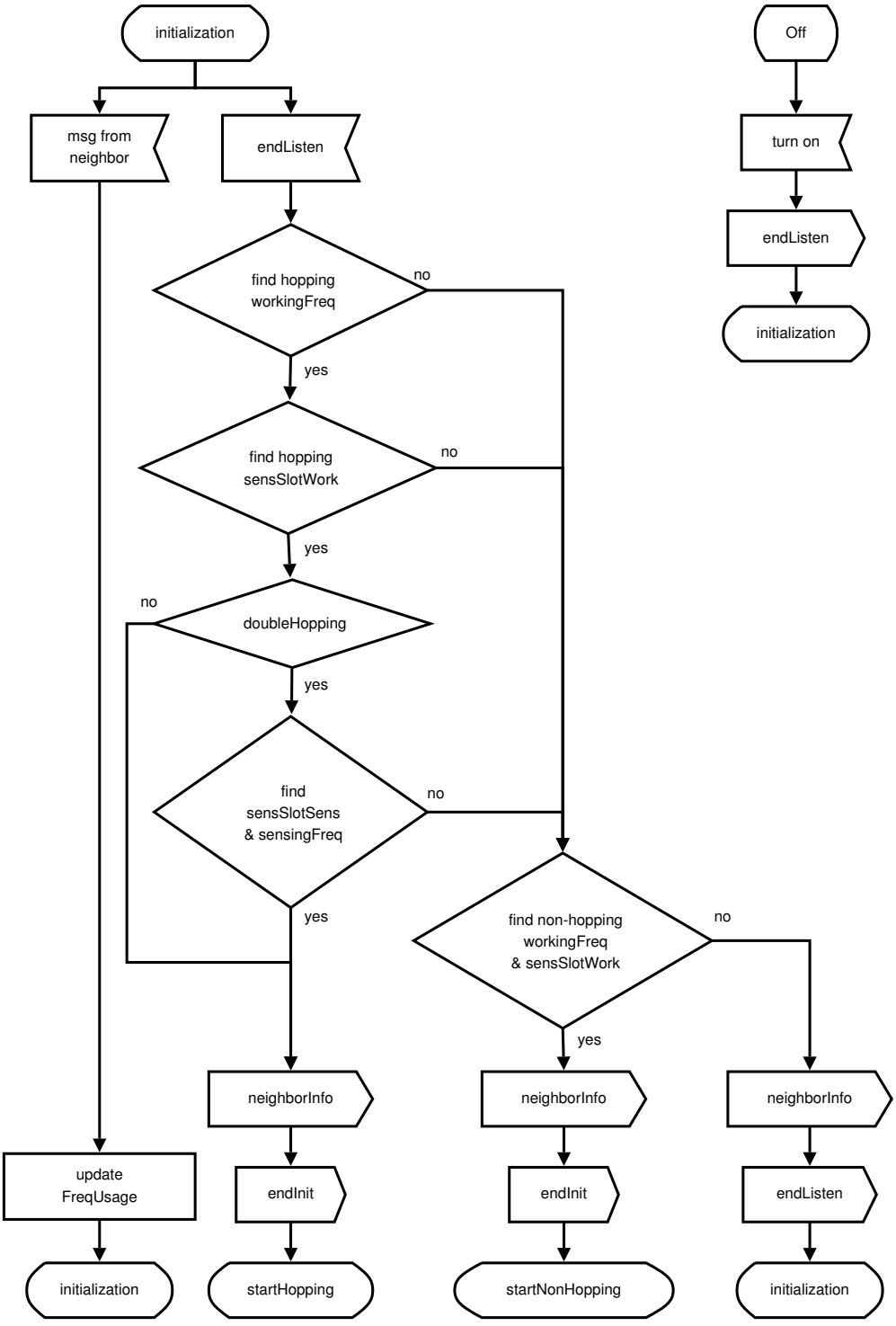


Figure A.1: `initialization` state

Hopping initialization

After having found a working frequency, the cell has to find a sensing slot for its working frequency (N_{wf}). The assumption is, that neighbors are not allowed to sense their working frequency (and thus hop to a new frequency) at the same time. This ensures a collision-free operation of neighboring CR cells. If no N_{wf} can be found, the cell tries to start operation in non-hopping mode as described in Section A.2.

In the DH mode, the cell additionally needs a sensing frequency and a sensing slot for the sensing frequency (N_{sf}). As shown in Chapter 3, neighboring cells should ideally use the same sensing frequency as well as the same slot for sensing the sensing frequency (N_{sf}). Thus, the cell first tries to use a sensing frequency and N_{sf} already used by its neighbors. Only if this is not possible it starts a new sensing frequency or chooses a different N_{sf}. If either no sensing frequency or no N_{sf} was found the cell tries to start operation in non-hopping mode as described in Section A.2.

After choosing the frequencies and sensing slots a `neighborInfo` message with the state set to `startHopping` is sent to announce the intended frequency usage to the neighbors. An `endInit` timer is set to the N_{wf}, where the initialization phase will be over and the cell starts to use the working frequency for the first time. The cell switches to the `startHopping` state shown in Figure A.2 and described in Section A.3.

Non-hopping initialization

If no N_{wf} (N_{sf}) was found for the working frequency (sensing frequency) or the cell cannot find a working or sensing frequency, it has to try to operate in non-hopping mode.

For the non-hopping mode, the cell has to find a working frequency in the non-hopping frequency range not used by any other neighboring cell. If successful, it has to find a N_{wf} not used by any *non-hopping* neighbor-cell.

On success, the cell switches to the `startNonHopping` state (shown in Figure A.3 and described in Section A.3) and sets the `endInit` timer to the N_{wf} as in the hopping initialization. It also broadcasts a `neighborInfo` message to announce the intended frequency usage. If no working frequency or N_{wf} can be found, the cell has to stay in the `initialization` state and reset the `endListen` timer.

A.3 `startHopping` / `startNonHopping` state

In the `startHopping` state and `startNonHopping` state potential collisions in the working and sensing frequency or sensing slots due to simultaneously starting cells are resolved. If a cell receives a `neighborInfo` message with conflicting frequencies or sensing slots, the start times of the two cells are compared. The cell with the smaller start time wins. If the start times are equal, the random number is used, where the cell with the smaller random number wins. The cell which looses, has to resolve the conflict (i.e. find new frequencies and / or sensing slots) and broadcast its new information in a `neighborInfo` message. The state diagrams are shown in Figure A.2 and Figure A.3 respectively.

If a *jump announcement* (`jumpAnnounce`) message or *new sensing frequency* (`newSensFreq`) message is received which contains conflicting frequencies or sensing slots, the conflicts have

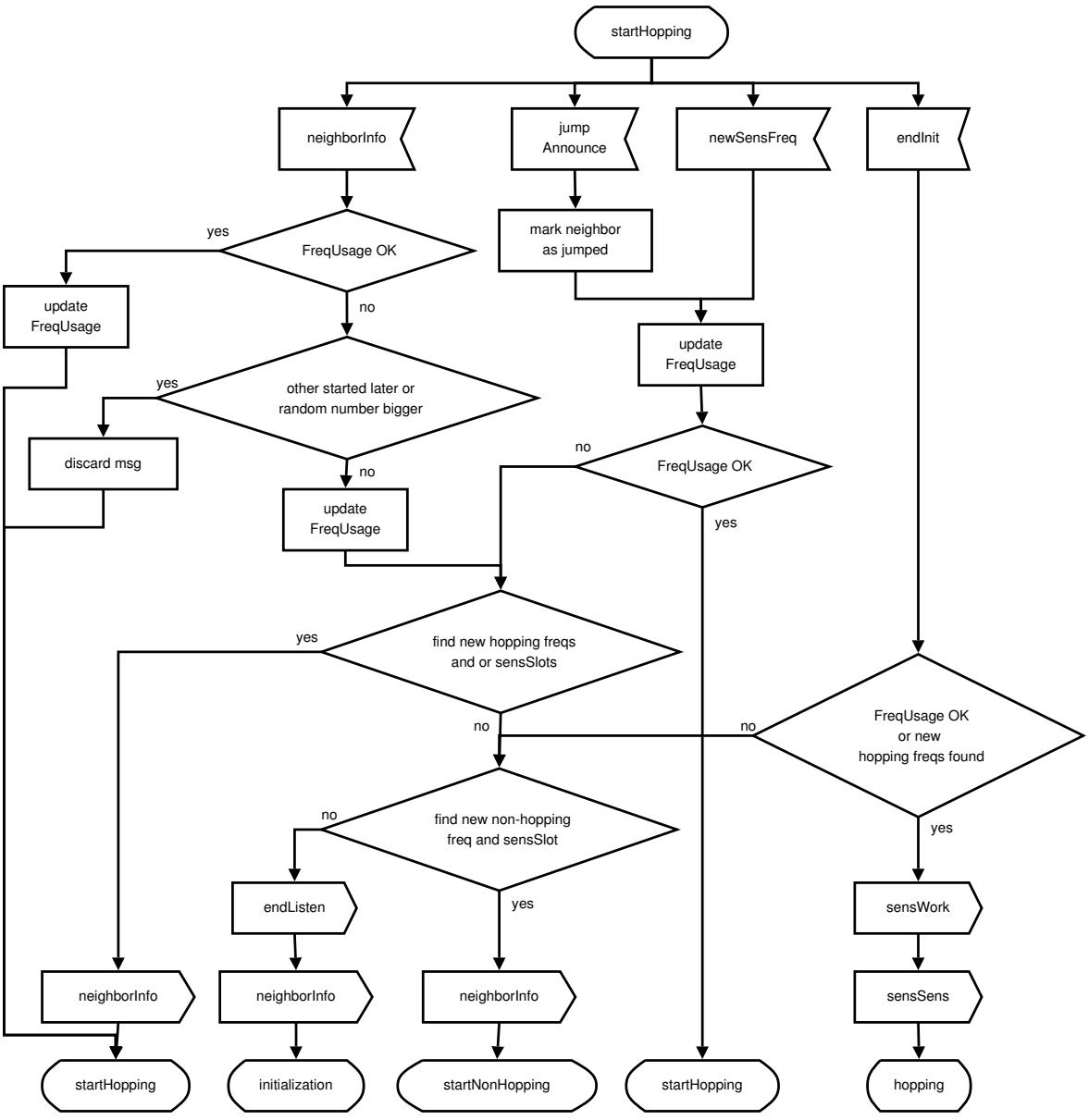


Figure A.2: `startHopping` state

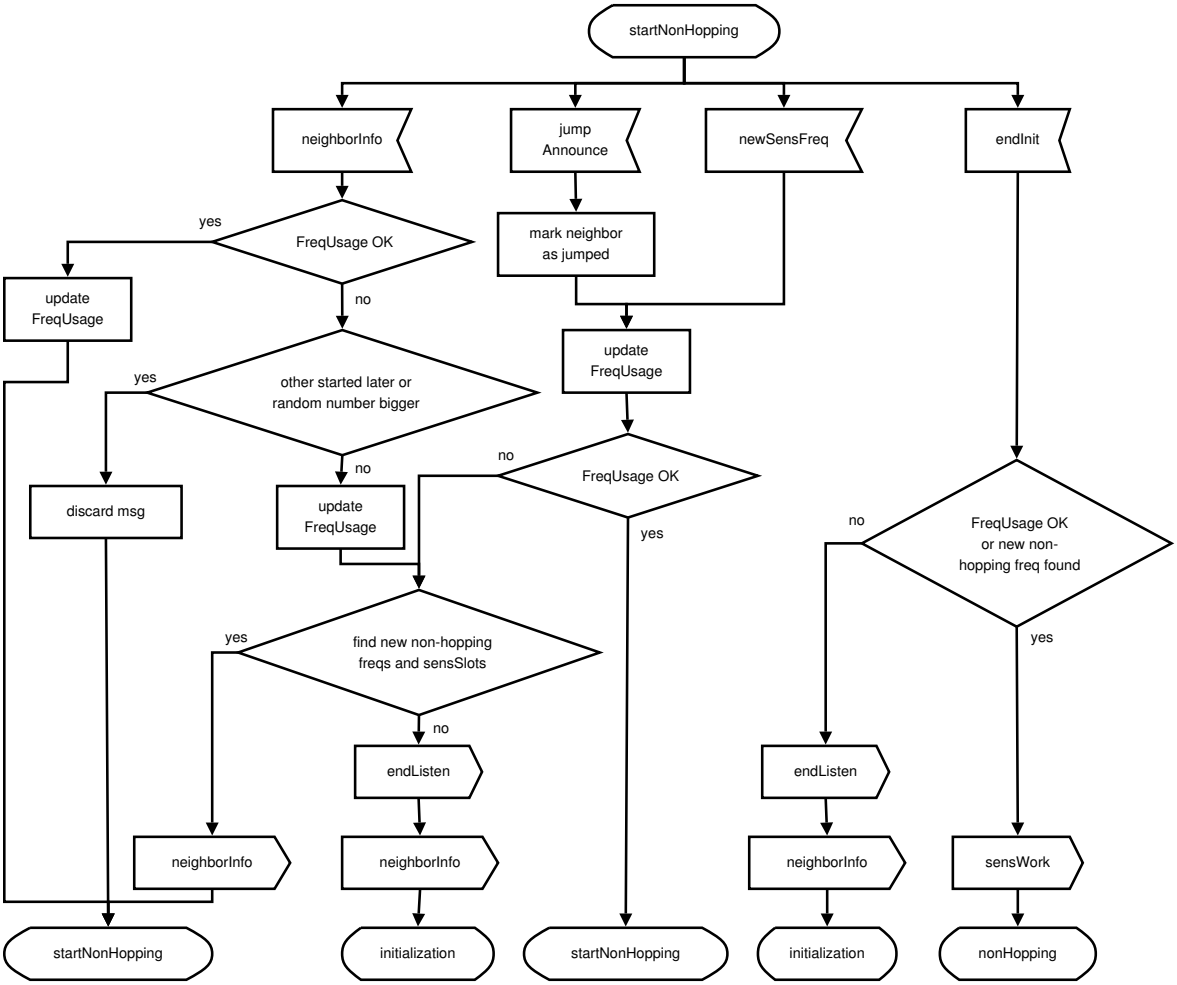


Figure A.3: `startNonHopping` state

to be resolved by the cell. Cells that currently are in the initialization phase have to integrate themselves into the existing frequency usage of cells already in the `hopping` / `nonHopping` state. A received `jumpAnnounce` or `newSensFreq` message indicates that the originator is already in the `hopping` / `nonHopping` state, which is why the conflict has to be resolved by the cell that currently is in the initialization phase.

The `endInit` timer expires at the beginning of the slot for sensing the working frequency (N_{wf}). The chosen frequencies have to be sensed again, to assure, that no PU appeared in the meantime. If the frequency usage is still OK or an alternative working frequency can be found, the cell switches to the `hopping` / `nonHopping` state and performs its first jump as described in Chapter A.4 and Chapter A.6 respectively. Finally, the `sensWork` timer and in DH mode also the `sensSens` timer have to be set.

If the frequency usage is not OK, i.e. some frequency cannot be used anymore, the cell cannot start operation. In the `startHopping` state, the cell tries to find a non-hopping working frequency and N_{wf} and – if successful – resets the `endInit` timer and switches to the `startNonHopping` state. If no non-hopping frequency and N_{wf} can be found the cell has to go back to the `initialization` state. If the cell is in the `startHonHopping` state and the frequency usage is not OK the cell also has to go back into the `initialization` state. In any case a `neighborInfo` message is broadcasted to announce the new state to the neighbors.

A.4 `hopping` state

Upon reception of any message from a neighbor (`neighborInfo`, `jumpAnnounce`, or `newSensFreq` message) the frequency usage is checked for conflicts. If no conflict is found the frequency usage is updated as previously described. If the received message is a `jumpAnnounce` message, the respective cell will be marked as already jumped in the `jumpedList`. This list serves for assuring that all neighboring cells have jumped before the cell performs its jump. Note, that strictly speaking the check for conflicts as well as the `jumpedList` are not necessary in the current implementation. Since we assume reliable message exchange, no conflicts should occur in the `hopping` state. However, our implementation is already prepared to also handle message losses. Details can be found in Section A.4.

The operation in RH and DH mode once the `sensWork` timer expires is slightly different and explained in the following sub-sections. The state diagram for the `hopping` state is shown in Figure A.4.

Error handling

This section gives an idea of steps to follow in case a cell detects potential conflicts with the frequency usage of some neighboring cell. This can happen if control messages (`neighborInfo`, `jumpAnnounce`, or `newSensFreq` messages) get corrupted or lost.

A message received from a neighboring cell may contain a frequency usage conflicting with the own frequency usage. This means that either originator of that message did not receive all messages sent by the cell correctly, or that some previous message of that neighbor got lost. The conflict should tried be resolved. First, the cell should trying to contact the conflicting neighbor to raise and resolve the conflict. If this fails, the cell should try to intelligently avoid further conflict by means to be determined.

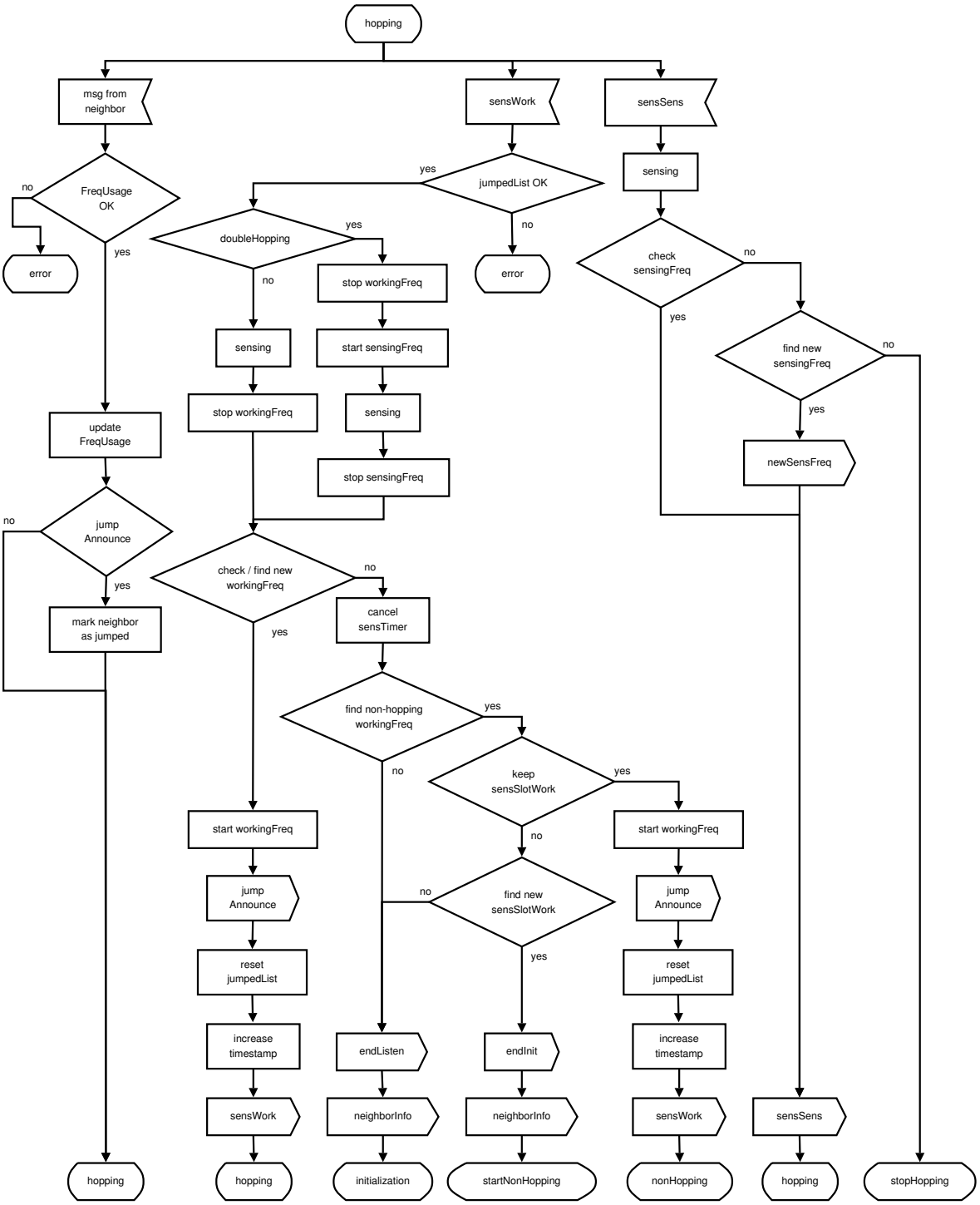


Figure A.4: `hopping` state

The `jumpedList` is another means to detect potential message losses. Once the `sensWork` timer expires, a cell has to assure that all neighboring cells in the `hopping` state have already jumped to a new frequency (`jumpedList OK` decision in the state diagram). If not all cells have jumped, some `jumpAnnounce` messages were not received correctly and thus the cell might not have the actual frequency usage of all neighbors. In this case the cell can try to guess the actual frequency usage of the respective neighbor and avoid these frequencies. Additionally, it should try to contact the neighbor to raise and resolve the issue.

Note, that the current implementation assumes reliable message transfer, so that all messages are always received correctly. Thus, the above described scenarios would result in an error and abortion of the simulation.

Revolver Hopping (RH)

Once the N_{wf} slot is reached, the cell tries to find a new working frequency by sensing a certain subset of potential working frequencies. Note that while sensing is performed, data tranmission continues on the current working frequency.

After t_{quiet} time units, i.e. one slot later, the sensing is finished and hopefully a new working frequency was found. The cell now stops operation on the current working frequency and jumps to the just validated new working frequency. The timestamp is increased (by t_{frame}) and the `jumpTimer` is set (to the current time plus t_{frame}). The `jumpedList` is reset and a `jumpAnnounce` message is broadcasted to announce the new working frequency to the neighbors.

If no new working frequency could be found, the cell tries to switch to the non-hopping mode. It tries to find a non-hopping frequency and on success, also a non-hopping N_{wf}. If the same N_{wf} as used before in the hopping mode can also be used in the non-hopping mode, the cell switches directly to the `nonHopping` state and jumps to the non-hopping working frequency. As in the normal hopping operation, the timestamp is increased, the `jumpTimer` is set, the `jumpedList` is reset, and a `jumpAnnounce` message is broadcasted.

If however, a non-hopping N_{wf} different from the hopping one was chosen, the cell has to switch to the `startNonHopping` state, set the `endInit` timer, and broadcast a `neighborInfo` message. The reason for this procedure is to avoid collisions due to concurrent jumps. If a non-hopping neighbor is using the same (non-hopping) N_{wf} as this (previously hopping) cell, it might happen that they choose the same working frequency for operation (as the selection is done concurrently). These collisions can only be avoided by going back to the `startNonHopping` state, where the cell can detect and resolve a potential collision.

If no non-hopping working frequency and / or N_{wf} could be found, the cell cannot continue operation and has to switch back to the `initialization` state and set the `endListen` timer. A `neighborInfo` message is broadcasted to announce the new status of the cell.

Double Hopping (DH)

In the DH mode, operation on the working frequency is stopped as soon as the N_{wf} slot is reached, and the cell jumps to the sensing frequency. Now, the working frequency is sensed for one slot (t_{quiet}) in order to assure no PU appeared. If the working frequency can still be used, the cell jumps back to the working frequency after the sensing is finished. If not, but

a new working frequency was found, the cell jumps to the new working frequency. As in the RH mode the timestamp is increased, the `jumpTimer` is set, the `jumpedList` is reset, and a `jumpAnnounce` message is broadcasted.

If no hopping working frequency could be found, the cell cancels the `sensSens` timer and tries to switch to the non-hopping mode as described above for the RH. If operation in non-hopping mode is not possible either, the cell has to switch back to the `initialization` state and set the `endListen` timer. A `neighborInfo` message is broadcasted to announce the new status of the cell.

In the DH mode, the sensing frequency also has to be sensed periodically. This is always done during the N_{sf} slot. If the sensing frequency cannot be used anymore, the cell tries to find a new sensing frequency and – upon success – has to broadcast the new frequency usage in a `newSensFreq` message. If the old sensing frequency is still usable no message is broadcasted. In both cases the `sensTimer` is set (to the current time plus t_{frame}).

If the old sensing frequency is not usable anymore and no new sensing frequency could be found, the cell has to switch to the `stopHopping` state until the next sensing period for the working frequency. The `sensTimer` is not set again in this case.

A.5 `stopHopping` state

The `stopHopping` state only exists for the DH mode. A cell has to switch to the `stopHopping` state if no new sensing frequency can be found as described above. Incoming messages are handled exactly the same way as in the `hopping` state described in Section A.4. The state diagram for the `stopHopping` state is shown in Figure A.5.

If the N_{wf} slot is reached, transmission on the working frequency has to be stopped. Since no valid sensing frequency was found, the cell cannot switch to the sensing frequency to continue data transmission and, thus, has to interrupt payload communication. However, the cell tries to switch to the non-hopping mode i.e. tries to find a non-hopping working frequency and sensing slot as described in Chapter A.4. If that is not possible the cell has to go back to the `initialization` state, set the `endListen` timer, and broadcast a `neighborInfo` message.

A.6 `nonHopping` state

Incoming message handling in the `nonHopping` state is the same as in the `hopping` state described earlier. The frequency usage is checked, updated, and the neighbor is marked as jumped if the message was a `jumpAnnounce`. The state diagram for the `nonHopping` state is shown in Figure A.6.

Once the N_{wf} slot is reached, data-tranmission has to be stopped on the working frequency in order to perform sensing. During the sensing, the cell not only checks whether it still can use its old (non-hopping) working frequency but also tries to find a hopping working frequency in order to go back to the hopping mode as it is the preferable operation mode. If a hopping working frequency was found the cell tries to keep its non-hopping N_{wf} slot. If it cannot keep the sensing slot, it tries to find a new one. Depending on whether or not the N_{wf} can be kept, the cell switches to the `startHopping` or directly to the `hopping` state, provided a N_{sf}

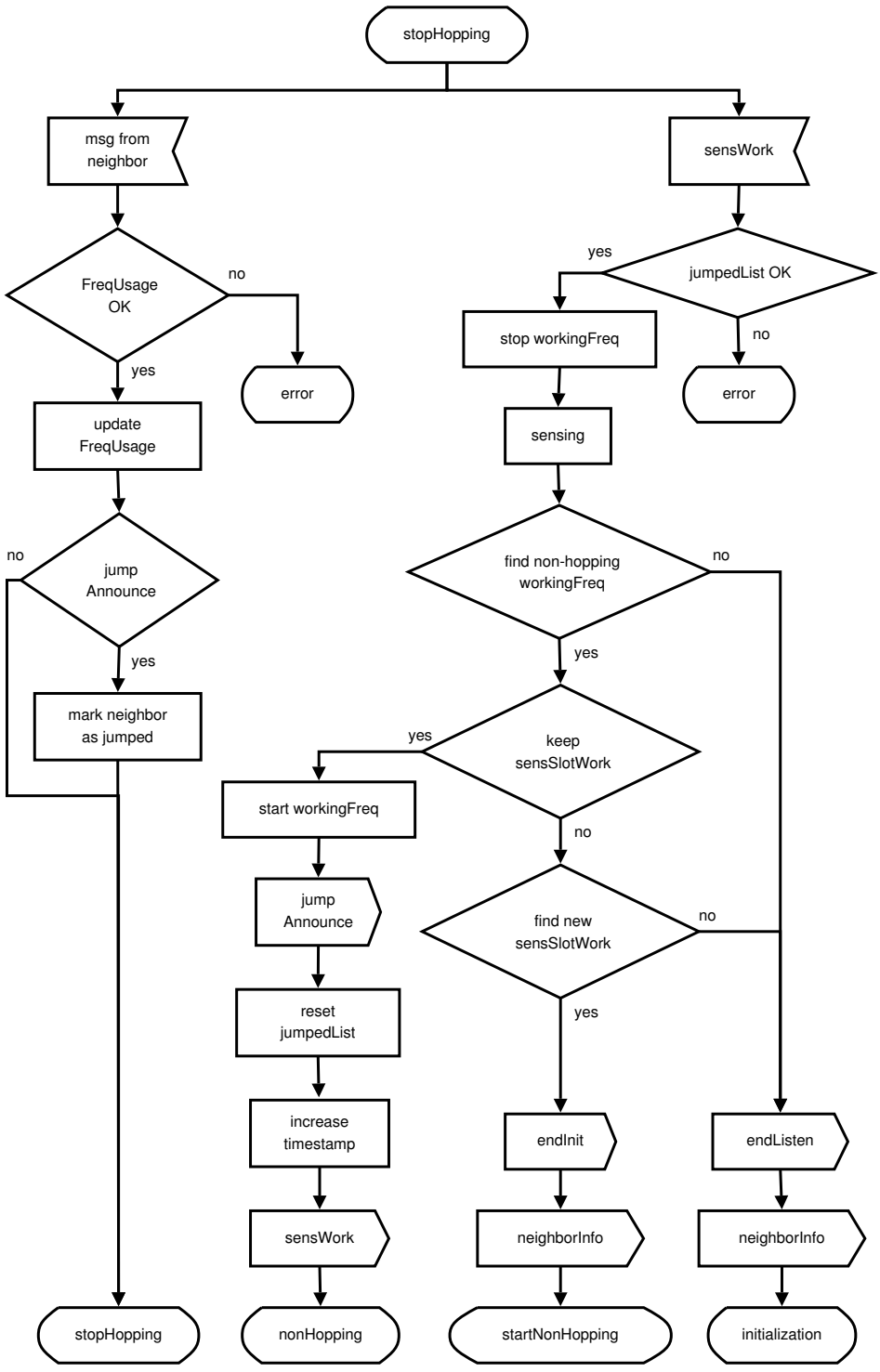


Figure A.5: `stopHopping` state

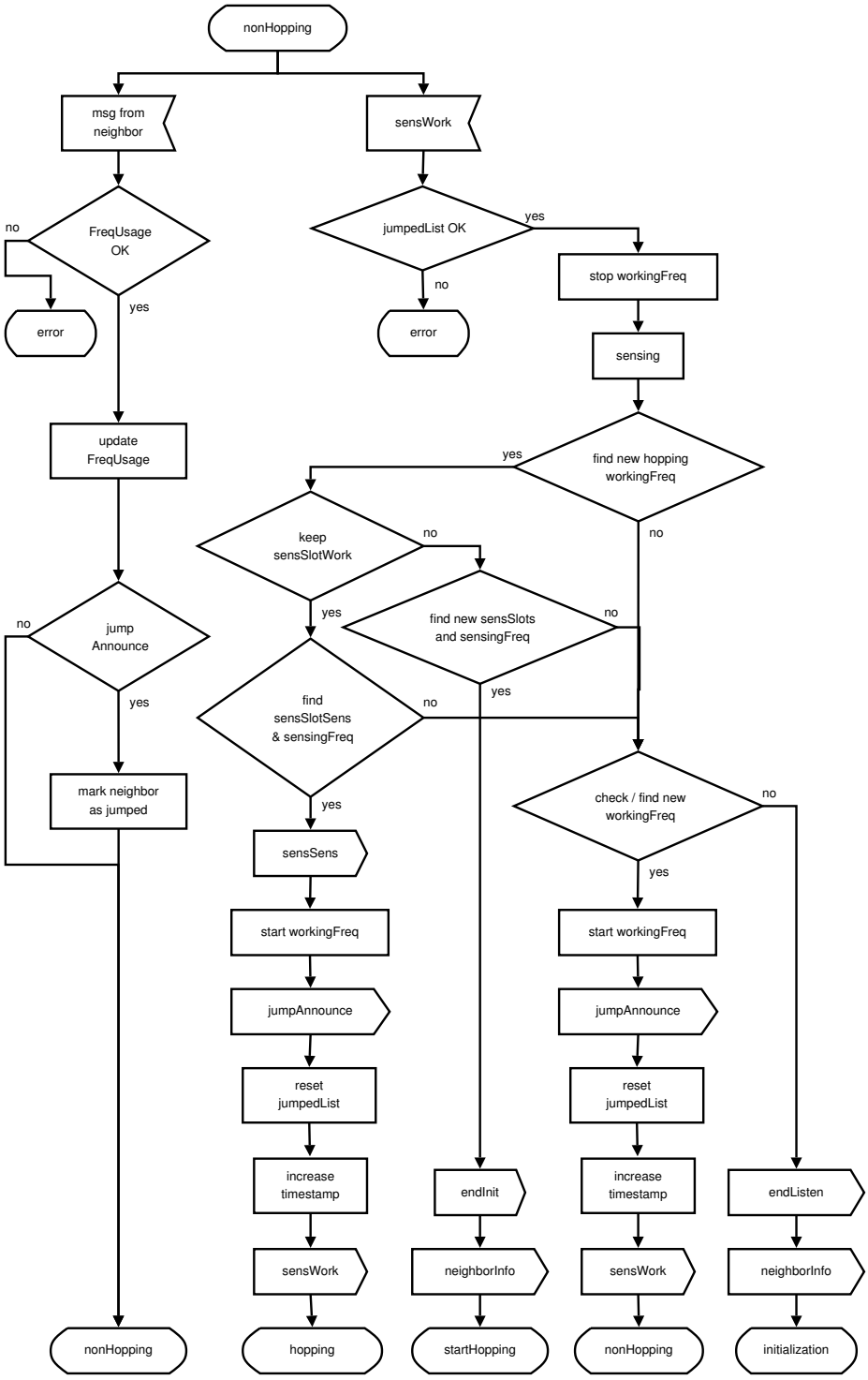


Figure A.6: Non hopping state

slot and sensing frequency was found. The reason for differentiating between keeping the old N_{wf} or choosing a new one is the same as described for the `hopping` state.

If the cell cannot switch back to the hopping mode, it has to revalidate its old non-hopping working frequency or find a new one. Upon success it starts operation on the working frequency. The timestamp is increased (by t_{frame}) and the `jumpTimer` is set (to the current time plus t_{frame}). The `jumpedList` is reset and a `jumpAnnounce` message is broadcasted to announce the new working frequency to the neighbors.

If the old working frequency is not usable anymore and no new working frequency could be found, the cell has to go back to the `initialization` state, set the `endListen` timer, and broadcast a `neighborInfo` message.

A.7 Time synchronization

In order to synchronize the hopping patterns of neighboring nodes, the timestamps need to be synchronized. To do so, the timestamp of every received message is compared with the own timestamp. The timestamps are compared modulo t_{frame} in order to determine which one has the earlier frame start. If the timestamp contained in the message has an earlier frame start, it is copied, otherwise it is ignored.

The N_{wf} slot has to be translated to the new timestamp, i.e. the absolute time for sensing the working frequency does not change, only the N_{wf} slot due to the new timestamp.

In DH mode the N_{sf} slot is copied from the neighbor with the smaller timestamp, if that slot is still available, i.e. the absolute time for sensing the sensing frequency changes. On success a `neighborInfo` message is broadcasted.

The reason for trying to adapt the N_{sf} slot from the neighbor is that with this approach the slots used for sensing the sensing frequency are reduced. As explained in Chapter 3, ideally all neighbors should perform sensing of the sensing frequency at the same time. The more different N_{sf} slots are used, the less neighbors can fit into the hopping structure.

However, it might happen that the N_{sf} slot from the neighbor is not usable, as it is used as a N_{wf} slot by some other neighbor. In this case the old N_{sf} slot has to be translated in the same way as the N_{wf} slot.

Appendix B

Acronyms

AFH	**A**daptive **F**requency **H**opping
BS	**B**ase **S**tation
CCI	**C**o-**C**hannel **I**nterference
CDH	**C**ommon **D**ouble **H**opping
CR	**C**ognitive **R**adio
DFH	**D**ynamic **F**requency **H**opping
DFHC	**D**ynamic **F**requency **H**opping **C**ommunity
DH	**D**ouble **H**opping
DFA	**D**istributed **F**requency **A**llocation
DHA	**D**istributed **H**opping **A**pproach
DLF	**D**istributed **L**argest **F**irst
FAP	**F**requency **A**ssignment **P**roblem
IDH	**I**ndividual **D**ouble **H**opping
GSM	**G**lobal **S**ystem for **M**obile Communications
LIP	**L**inear **I**nteger **P**rogram
MAL	**M**obile **A**llocation **L**ist
OFA	**O**ptimal **F**requency **A**llocation
PER	**P**acket **E**rror **R**ate
PMF	**P**robability **M**ass **F**unction
PU	**P**rimary **U**ser

QoS	**Q**uality **o**f **S**ervice
RH	**R**evolver **H**opping
SU	**S**econdary **U**ser
TDMA	**T**ime **D**ivision **M**ultiple **A**ccess
WLAN	**W**ireless **L**ocal **A**rea **N**etwork
WRAN	**W**ireless **R**egional **A**rea **N**etwork

$|V|$ total number of CR cells in the network

F_{tot} total number of frequencies available for secondary communication

F_{nh} first non-hopping frequency

t_{max} maximal interference time

t_{sens} sensing time [ms]

t_{quiet} quiet time [ms]

t_{data} data transmission time [ms]

t_{frame} frame time [ms]

N_q number of quiet times (t_{quiet}) per data transmission time (t_{data})

N_{wf} sensing slot for the working frequency

N_{sf} sensing slot for the sensing frequency

F_{w} working frequency

F_{s} sensing frequency

Bibliography

[1] L. Chu, W. Hu, G. Vlantis, J. Gross, M. Abusubaih, D. Willkomm, and A. Wolisz, "Dynamic frequency hopping community," online, IEEE 802.22 Working Group, Technical proposal submitted to IEEE 802.22 WG 22-06-0113, Jun. 2006.

[2] W. Hu, D. Willkomm, L. Chu, M. Abusubaih, J. Gross, G. Vlantis, M. Gerla, and A. Wolisz, "Dynamic frequency hopping communities for efficient IEEE 802.22 operation," *IEEE Commun. Mag., Special Issue: "Cognitive Radios for Dynamic Spectrum Access"*, vol. 45, no. 5, pp. 80–87, May 2007.

[3] A. Eisenblaetter and A. Koster, "FAP web page," http://fap.zib.de, 2000.

[4] K. H. Rosen, *Discrete Mathematics and Its Applications*. McGraw-Hill Higher Education, 1998.

[5] D. Hollos, D. Willkomm, J. Gross, and W. Hu, "Centralized vs. distributed frequency assignment in frequency hopping (cognitive radio) cellular networks," Telecommunication Networks Group, Technische Universität Berlin, Tech. Rep. TKN-07-007, Dec. 2007.

[6] P. Björklund, P. Värbrand, and D. Yuan, "Optimal frequency planning in mobile networks with frequency hopping," *Computers and Operations Research*, vol. 32, pp. 169–186, 2005.

[7] J. Moon, L. Hughes, and D. Smith, "Assignment of frequency lists in frequency hopping networks," *IEEE Trans. Veh. Technol.*, vol. 54, no. 3, pp. 1147–1159, 2005.

[8] Z. Kostic, I. Maric, and X. Wang, "Fundamentals of dynamic frequency hopping in cellular systems," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 11, pp. 2254–2266, Nov. 2001.

[9] M. Cho-Hoi Chek and Y.-K. Kwok, "On adaptive frequency hopping to combat coexistence interference between bluetooth and IEEE 802.11b with practical resource constraints," in *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks*, 2004, pp. 391–396.

[10] B. Zhen, Y. Kim, and K. Jang, "The analysis of coexistence mechanisms of bluetooth," in *Preceedings of the 55th IEEE Vehicular Technology Conference, 2002. (VTC Spring 2002)*, vol. 1, May 2002, pp. 419–423.

[11] H. Yomo, P. Popovski, H. Nguyen, and R. Prasad, "Adaptive frequency rolling for coexistence in the unlicensed band," *IEEE Trans. Wireless Commun.*, vol. 6, no. 2, pp. 598–608, 2007.

[12] A. Mishra, V. Shrivastava, D. Agrawal, S. Banerjee, and S. Ganguly, "Distributed channel management in uncoordinated wireless environments," in *Proceedings of the 12th annual international conference on Mobile computing and networking (MobiCom 2006)*. New York, NY, USA: ACM, 2006, pp. 170–181.

[13] "IEEE 802.22 working group," online. [Online]. Available: http://www.ieee802.org/22

[14] B. Sanso and P. Soriano, Eds., *Telecommunications Network Planning*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[15] A. Koster, "Frequency assignment: Models and algorithms," Ph.D. dissertation, Universiteit Maastricht, 1999.

[16] C. Peng, H. Zheng, and B. Y. Zhao, "Utilization and fairness in spectrum assignment for op-

portunistic spectrum access," *Mobile Networks and Applications*, vol. 11, no. 4, pp. 555–576, 2006.

[17] S. Sengupta, S. Brahma, M. Chatterjee, and S. Shankar N, "Enhancements to cognitive radio based IEEE 802.22 air-interface," in *Proceedings of the IEEE International Conference on Communications, 2007 (ICC '07)*, 24-28 June 2007, pp. 5155–5160.

[18] IEEE802.22, "IEEE P802.22/D0.1 draft standard for wireless regional area networks part 22: Cognitive wireless ran medium access control (MAC) and physical layer (PHY) specifications: Policies and procedures for operation in the TV bands," IEEE 802.22 Working Group, Draft Standard, 2006.

[19] D. Willkomm, M. Bohge, D. Hollos, and J. Gross, "Double hopping: A new approach for dynamic frequency hopping in cognitive radio networks," Telecommunication Networks Group, Technische Universität Berlin, Tech. Rep. TKN-08-001, Jan. 2008.

[20] M. Kubale and L. Kuszner, "A better practical algorithm for distributed graph coloring," in *Proceedings of the IEEE International Conference on Parallel Computing in Electrical Engineering 2002 (PARELEC 02)*. IEEE, 2002, pp. 72 – 76.

[21] J. Culberson, A. Beacham, and D. Papp, "Hiding our colors," in *CP'95 Workshop on Studying and Solving Really Hard Problems*, Cassis, France, 1995, pp. 31–42.

[22] S. ILOG, "CPLEX solver," 2007, (accessed 04. Mai 2007). [Online]. Available: http://www.ilog.fr/products/cplex/

[23] A. Varga, *OMNeT++ Discrete Event Simulation System*. [Online]. Available: http://www.omnetpp.org/doc/manual/usman.html

[24] "Mobility framework (MF) for simulating wireless and mobile networks using OMNeT++." [Online]. Available: http://mobility-fw.sourceforge.net/