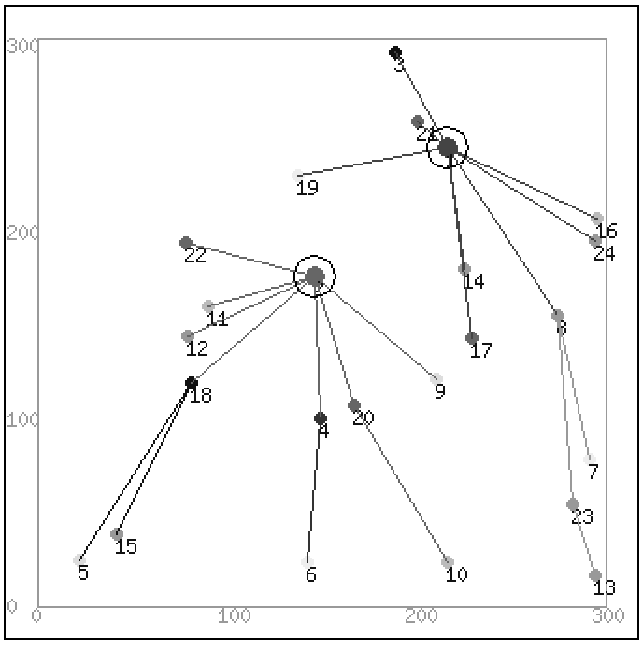
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze



Channel Allocation for IEEE 802.16 Mesh Networks

Till Kleisli

Master's Thesis MA-2006-24
April 2006 until October 2006
Advisors: Dr. Ulrich Fiedler, Rainer Baumann
Supervisor: Prof. Dr. Bernhard Plattner

Abstract

In this thesis, we investigate the problem of channel allocation in a static multi-hop wireless mesh network employed for telephony in public safety scenarios. We assume that each mesh router is equipped with two IEEE 802.16-2004 radio interfaces. These interfaces can be switched to one of 12 orthogonal channels. The objective is to find a set of channel allocations for the interfaces in the network that minimize the network interference and fairly distribute capacity.

We evaluate three classes of algorithms for channel allocation: Random allocation, greedy allocation and merge allocation with numeric calculations on graphs as well as with simulations in QualNet.

We study the following scenarios for distribution of mesh routers: Random placement of nodes with four different numbers of nodes and grid placements of nodes also with four different number of nodes.

The numeric calculations approach reveals that all greedy algorithms perform well and limitations in capacity are mainly due to a suboptimal topology construction.

The QualNet simulation shows that networks build of routers with IEEE 802.16-2004 interfaces may not have sufficient capacity for the studied public safety scenarios.

Contents

Chapter 1

Introduction

This chapter gives an overview over the contents of this thesis, the problem and its context.

1.1 Context

1.1.1 IEEE 802.16 and mesh networks

Wireless mesh networks create connectivity over multiple wireless hops. With IEEE 802.16 technology maturing within the next 2-3 years, wireless mesh networks become an interesting alternative to wired access in metropolitan areas. Moreover, communication systems based on wireless mesh networks are of particular interest to public safety applications, e.g. to create network connectivity right after a disaster such as an earthquake or a tsunami.

The IEEE 802.16 standard is also often referred to as WiMax. The WiMAX Forum [1] is an industry-led, non-profit corporation formed to promote and certify compatibility and interoperability of broadband wireless products based on IEEE 802.16 and ETSI HiperMAN wireless MAN standards.

1.1.2 The TOWN project

In the TOWN project, we search for algorithms that can be employed in public safety applications where telephony is the primary application. Thus, traffic demand is fixed and predictable. Since interface cards working in full mesh mode will likely not be available within the next 2-3 years, the TOWN project heads toward developing algorithms and protocols for frequency allocation assuming that network nodes are equipped with two or more interfaces. The first interface works in base station mode to offer network connectivity to multiple other nodes. The second and further interfaces work in subscriber station mode to connect the node to the network. This interface configuration leads to point-to-multi point communication relations.

Channel allocation in a TOWN network follows the following principle. All nodes subscribed to a node B, use one channel to communicate with node B, this channel is determined by the base station interface of B. That means every node subscribed to node B tunes its subscriber station interface to the same channel the base station interface of node B is tuned to. The subscriber station interface of node B then uses a different channel to subscribe to another node C. (see Fig. 1.1)

Nodes in a TOWN mesh network are either routers or gateways. Routers are equipped with a base station interface and a subscriber station interface. Gateways are also equipped with a base station interface and additionally offer connectivity

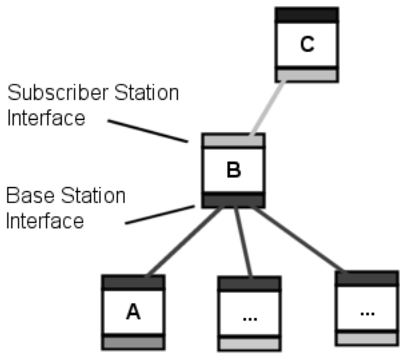


Figure 1.1: Example of a network constructed with multiple radio nodes. Node A is subscribed to node B, i.e. the subscriber station interface of node A is on the same channel as the base station interface of node B, and node B is subscribed to node C.

to the wired telephone network. Each router needs to have a connection to any gateway to be able to establish a telephone line.

The focus of the TOWN project is on scenarios where network nodes are stationary, i.e. do not move. In addition, in this thesis we focus on the frequency allocation problem for a given mesh network spanning tree topology for the two interface case and assume that no nodes join or leave the network.

For communication, 12 orthogonal channels are available i.e. there are 12 channels not interfering with each other.

1.2 Problem Description

Given a set of mesh routers and gateways, we have to construct a mesh network that meets the requirements of the telephony application. Every router must have a connection to a gateway and a fixed available bandwidth to and from this gateway.

This problem is NP-hard [2]. Even if we assume that there is no interference in the network, we can't compute the optimum in polynomial time. There are approaches to solve this problem joint, as in [3] or [4], but we attempt to decouple the problem and optimize the parts of it.

We divide the problem into the two subproblems topology construction and channel allocation. This thesis has its main focus on channel allocation for a given topology, but as the topology construction is also part of the problem, we also considered the problem of topology construction for a given set of nodes as a minor part of this thesis.

1.2.1 Topology Construction

The task of topology construction is to connect the mesh nodes. On a given set of routers and gateways, we need to build communication trees with the gateways as roots. (Fig. 1.2)

In topology construction we primarily have to ensure that every router is part of a tree. Secondarily we need to balance the number of routers per tree to ensure a certain bandwidth to the gateway for each node. As the traffic demand per node is fixed and the link capacity limited, more than necessary routers per gateway restrict the capacity.

To solve the topology construction problem, we have no position information. The only information we get, is what other nodes are within communication range

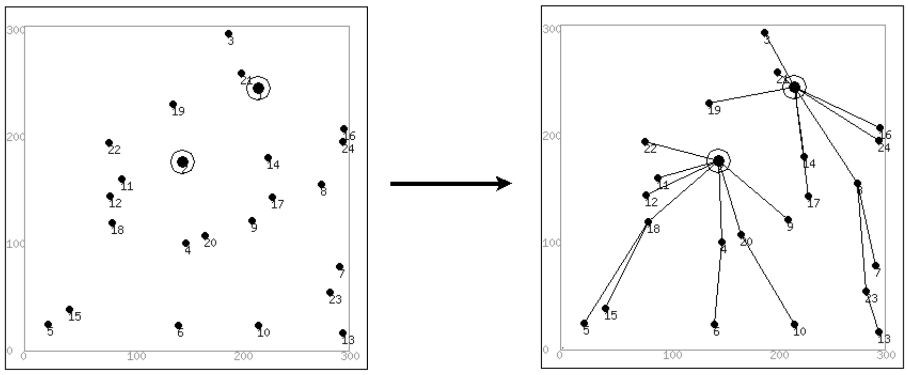


Figure 1.2: Topology construction leads from a set of routers and gateways to a topology of trees with the gateways as roots. Gateways are marked by a circle.

and additional information about these nodes.

1.2.2 Channel Allocation

The task of channel allocation is to assign each communication link a channel, i.e. tune both endpoint interfaces to the same channel, on a given topology. (Fig. 1.3) For our network model, it is sufficient to allocate each base station interface a channel, as the channels of subscriber station interfaces, and thus also the channels of the communication link, are determined by the channel of their base station interface. (Fig. 1.1)

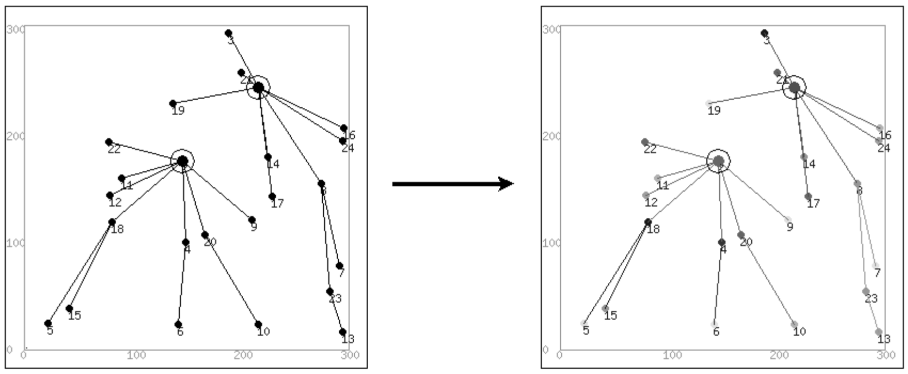


Figure 1.3: Channel allocation leads from a topology to a network with a channel assigned to each communication link. Gateways are marked by a circle.

With this definition of the problem, allocating channels to base station interfaces or to nodes respectively, as each node has one base station interface, we are already close to a graph coloring problem. The classical graph coloring colors the nodes of a graph with a minimal number of colors, in a way that no adjacent nodes have the same color (see also [5]). But our problem is more complicated, as we have to deal with continuous interference values, and not only binary link or no-link. Nonetheless we try to solve the channel allocation problem using also classical graph coloring principles like Greedy, adapting the algorithms for our problem.

To allocate the channels, an algorithm only has the topology information and interference values it can get by measurements at each node. The output of a channel allocation algorithm, the channel allocation scheme, should also meet the traffic demands of each node. Therefore it needs to minimize the interferences by distributing the channels in a clever way.

1.3 Algorithms

For topology construction we consider a simple algorithm which always connects the node farthest from any gateway next. Additionally we implemented a heuristic to balance the number of routers per gateway.

For channel allocation, we consider three groups of algorithms. A random allocation as baseline for other algorithms. Greedy algorithms, which allocate channels iteratively and always allocate the stepwise optimal channel. And a merge algorithm which starts by assigning each node a unique channel and iteratively merging two channels until the number of desired channels is reached.

1.4 Method

To evaluate the algorithms for the two subproblems we create different node placements on which we construct a topology and allocate channels on this topology. Then we use numeric calculations on graphs and simulation as complementary evaluation methods to evaluate the channel allocations.

For numeric calculations, we compute interference values in the network according to formulas and capacities of links based on these interference values.

For simulations, we use QualNet, a commercial network simulator. In the simulator, we simulate traffic on the mesh network and evaluate the throughput and the packet error ratio reached for each node. These values can then be compared to the results by the calculations.

We created scenarios with different numbers of nodes and an according number of gateways. The first part of scenarios consists of four scenarios with 12, 24, 36 and 48 randomly placed nodes, with one node out of 12 chosen as gateway. The second part of scenarios consists of four grid scenarios, where nodes are placed on a grid, in a way they can only communicate with their horizontal and vertical neighbors. The grid sizes evaluated are 5x5 nodes with two gateways, 6x6 nodes with three gateways, 7x7 nodes with four gateways and 8x8 nodes with six gateways.

Of the eight scenario classes we created five instances each. For random scenarios all nodes are placed randomly for the grid scenarios only the positions of the gateways were chosen randomly for each instance. Then we computed a topology for each of these instances and let different channel allocation algorithms compute channel allocation schemes.

These allocation schemes were evaluated using the following metrics: The bandwidth available to a router (capacity), the noise on the channel the base station interface of a node is tuned to (interference) and the balance of distribution of traffic on channels (fairness).

1.5 Major Findings

From the calculations, we can deduce, greedy algorithms are an efficient way to compute good channel allocation schemes. All three evaluated greedy algorithms show a similar performance concerning the minimal capacity of any node or the average capacity of all the nodes. Observed capacity limitations are mainly due to a suboptimal topology construction.

Regarding the simulation in QualNet, we encountered problems with the throughput. We couldn't achieve the throughput demands of the routers. Even for only one subscriber station and one base station. It has to be investigated, if that could also be a problem for a field test or if it is just a problem of the simulator.

1.6 Structure of thesis

The remaining document is organized as follows. In Chapter 2 we explain, how we evaluated the channel allocation algorithms, in Chapter 3 we present algorithms for channel allocation and topology construction, Chapter 4 gives the results of the evaluation, in Chapter 5 we present some related work and in Chapter 6 we discuss the results and suggest some future work.

Chapter 2

Methodology

This chapter describes the evaluation methods and evaluation criteria.

To evaluate channel allocation algorithms, we had different possibilities. Evaluation by proofing lower bounds for the results of the algorithms, which is very complex and often even not possible. Evaluation by testing the algorithms in the field, by setting up a network with multiple antennas, which is quite laborious and also too expensive for the moment. And last but not least, we can map the whole environment in which we want to apply the algorithms on a graph or into a simulator, to evaluate the algorithms theoretical in a almost practical environment.

The evaluation methods of choice finally were numeric calculations on graphs representing the network, and simulation as complementary evaluation methods.

2.1 Numeric Calculations on Graphs

For the numeric calculations, we consider a network as graph with the nodes as vertices and the communication links of the topology as edges. To evaluate the algorithms, we use the interference, capacity and fairness values as evaluation criteria.

As a first criteria, we look at the interferences between the nodes. The lower the interferences are, the more of the capacity of a link can be used for data transfer.

The capacity is also an important criteria, as in our context, every router has a need for a specific capacity of his route to the gateway. For this reason, we also look specially at the minimal capacity of any router in the mesh network.

Fairness as third evaluation criteria indicates how fair the channels are distributed. We assume an allocation to be fair, when every channel has about the same traffic load.

2.2 Evaluation Criteria

2.2.1 Interferences

We look at the interference each node senses on the channel its base station interface is tuned to. For this purpose we sum up all the interference values between a node and all other nodes having an interface on the same channel. Regarding the interferences, we compare the maximum interference value at any node, the average interference value at all nodes and the standard deviation of all the interference values.

Interference Model in General

For the interference calculation we have two different formulas, the free space model [6] and the two ray ground model. As the two ray ground model is not very realistic for short distances, we first calculate the crossover distance d_0 (Equation 2.1) where the two models have the same results and use the free space model for distances below or equal to d_0 and the two ray ground model for distances above d_0.

$$d_0 = \frac{4\pi \cdot h_1 \cdot h_2 \cdot f}{c} \tag{2.1}$$

where h_1 and h_2 are the heights of the two nodes, f is the frequency (5.8 GHz) and c is the speed of light.

This principle of applying the crossover distance is also used in the open source simulator ns-2 [7].

Free Space Model

The Free space model by H. T. Friis [6] considers the wave propagation in the free space on a sphere around the antenna.

The received power P_R as fraction of the transmitting power P_T is calculated as

$$P_R = \frac{c^2}{(4\pi \cdot d \cdot f)^2} \cdot P_T \tag{2.2}$$

where c is the speed of light, d is the distance of the two nodes and f is the frequency.

Two Ray Ground Reflection Model

The two ray ground reflection model describes the propagation of a wave over a plane. It considers the direct connection from the sender to the receiver plus the reflection of the wave on the plane.

Again we calculate the received power P_R as fraction of the transmitting power P_T as follows:

$$P_R = \frac{h_1^2 \cdot h_2^2}{d^4} \cdot P_T \tag{2.3}$$

where h_1 and h_2 are the heights of the two nodes and d is the distance between the two nodes.

2.2.2 Capacity

To get the capacity of a route from a node to his gateway, we add for every router an abstract traffic flow of 1 to every link on his route to the gateway. Then, we determine the collision domain of every link. The collision domain of a link consists of any other link, that has an interfering endpoint with any endpoint of the considered link.

For every link, we determine the sum of all traffic flows of its collision domain. Because the communication on one channel is done in time division multiplex (TDM), we assume, that the capacity of one link is diminished by all the other traffic flows on the same channel in interference range. As interference range, we assume three times the communication range, as it is also applied by [3] or [8].

For every router we then look for the link on his route to the gateway that has the maximum traffic flow in his collision domain, the so called "'bottleneck link"'. The capacity of a node is the link capacity divided by the number of traffic flows in the collision domain of its bottleneck link.

As evaluation criteria we can use the overall capacity of the network by summing up the capacities of all nodes. But, because we have different numbers of nodes, we better use the average capacity. Another important value is the minimum capacity, because in our context, all routers need to have a certain capacity (4Mbps) on their routes to the gateway.

2.2.3 Fairness

The fairness value f of a set of values x_i represents how equal they are. f is equal to 1, if all the values x_i are the same. The more the values x_i differ from each other, the smaller is the fairness value f.

To calculate fairness, we use Jain's Fairness Index [9]. For n values $[x_1, \ldots, x_n]$, the fairness index f is is defined as follows

$$f(x_1, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{2.4}$$

We calculate the fairness of the following distributions:

- the channels divided between the base station interfaces
- the channels divided between the links
- the channels divided between the traffic flows

2.3 Simulation

For the simulation part we evaluated several simulators like ns-2 [10], OpNet [11] and QualNet [12]. We chose QualNet, because as the only simulation environment QualNet supports the IEEE 802.16 MAC model, which we would have had to implement by ourselves in other simulators. In addition, QualNet uses a layered structure similar to that of the TCP/IP network protocol stack. Physical layer, Link (MAC) layer, Network layer, Transport layer and Application layer have well-defined APIs can be modified and implemented separately.

We set up simulations to simulate traffic and evaluate the packet error rate (PER) of different allocations. But we didn't reach the throughput required from a subscriber station to a base station, even for a configuration with just two nodes. Throughput from a base station to a subscriber station is no problem. Table 2.1 and Table 2.2 show the results of a test series we conducted with QualNet. The setup of the test consists of one subscriber station and one base station in distance 100m. The communication frequency is 5.8 GHz and the capacity of the physical layer 54Mbps. The subscriber station tries to send a CBR stream of 2Mbps to the base station.

We varied the packet size of the CBR packets as well as the minimal request bandwidth parameter of the WiMAX MAC layer. We evaluated the packet sizes sent by the physical layer and the throughput achieved at the base station.

packet size	128 kBps	256 kBps	300 kBps	400 kBps
128 B	2430 B 0.767 Mbps	486 B 0.153 Mbps	1296 B 0.409 Mbps	3402 B 1.073 Mbps
256 B	2320 B 0.818 Mbps	290 B 0.102 Mbps	1160 B 0.409 Mbps	3190 B 1.125 Mbps
512 B	2184 B 0.818 bps	 0.000 Mbps	1092 B 0.409 Mbps	3276 B 1.227 Mbps
1024 B	2116 B 0.819 Mbps	 0.000 Mbps	1058 B 0.410 Mbps	3174 B 1.227 Mbps

Table 2.1: Test series for QualNet, Part 1: The rows represent different sizes of CBR packets while the columns represent different values for the minimal request bandwidth by the WiMAX MAC layer. In each cell, the size of the packets sent by the PHY layer are indicated along with the throughput achieved at the base station.

packet size	425 kBps	450 kBps	475 kBps	500 kBps
128B	3888 B 1.226 Mbps	4374 B 1.379 Mbps	162 B 0.051 Mbps	810 B 0.256 Mbps
256B	3770 B 1.327 Mbps	4350 B 1.532 Mbps	290 B 0.102 Mbps	580 B 0.205 Mbps
512B	3822 B 1.431 Mbps	4368 B 1.635 Mbps	 0.000 Mbps	546 B 0.205 Mbps
1024B	3174 B 1.227 Mbps	4232 B 1.635 Mbps	 0.000 Mbps	 0.000 Mbps

Table 2.2: Test series for QualNet, Part 2

Chapter 3

Algorithms

In this chapter we review the algorithms studied in this thesis.

As we look at the channel allocation problem separated from the topology construction, we briefly present a topology construction algorithm and for channel allocation, we introduce three variants of a greedy algorithm, a merge algorithm and a random channel allocation.

3.1 Topology Construction

The topology construction algorithm takes as input all the node information we have, information about the gateways, the communication range and an upper bound for the number of routers allowed to be connected to a gateway.

All possible communication links, i.e distances below communication range, are computed.

The network is flooded with a message that contains a hop counter from each gateway. Each router stores the minimal hop distance to every gateway it can reach.

First all routers that have only one reachable gateway connect to that gateway. Then, while not all routers are connected, routers connect sequentially to a gateway ordered by decreasing maximum of the minimal hop distance to any gateway. By connecting a router to a gateway, all the routers on the way to this gateway automatically also get connected.

If the number of routers connected to a gateway exceeded 12 and another router wants to connect to this gateway, it checks all the other gateways the router can reach, if there is any gateway that has less than 12 nodes connected. If there is one or more such alternative gateway, the router connects to the gateway with the minimal hop distance. In this process, routers on the way to the alternative gateway that are already connected to another gateway also get reconnected to the alternative gateway.

3.2 Channel Allocation based on Random Allocation

As a baseline for comparisons of channel allocation schemes, we implemented a random channel allocation algorithm. Random allocation assigns each node uniformly distributed a random channel out of all channels.

3.3 Channel Allocation based on Greedy Algorithms

The idea of greedy algorithms is optimizing locally leads to a global good solution. In every step, greedy algorithms take the optimal solution for this step, in our context, they chose the least interfered channel to allocate to a node.

The three presented variants of greedy algorithms differ in the order in which the nodes are assigned a channel and in the mode channels are assigned to nodes.

3.3.1 Greedy Breadth First

For the first greedy variant we use the breadth first order of the nodes in the communication trees. As we can also see in Fig. 3.1, we first allocate a channel to all the gateways, then to all the routers with hop distance one to a gateway, then to all routes with hop distance two, and so on.

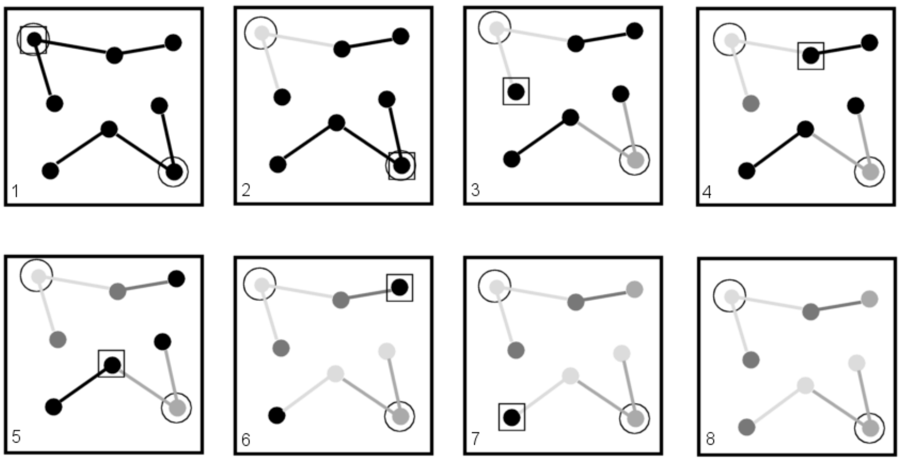


Figure 3.1: Example of the Greedy Breadth First algorithm with two gateways (marked with circles) and three channels. The node to be assigned a channel in the next step is marked with a square.

For each node to be allocated a channel, we look at the interferences from other already assigned base station and subscriber station interfaces on each channel. The channel with the least interference is chosen to be assigned to the base station interface of the actual node to be assigned a channel.

3.3.2 Greedy Most Interfered First

The greedy most interfered first algorithm is inspired by [13]. The channel of the next node to be assigned a channel is chosen the same way as in Greedy Breadth First, but the Greedy Most Interfered First algorithm uses a different method to determine the next node.

First, the average of all interference values on all nodes on all channels is computed. Then each node determines the number of channels with an interference value above the average value. The node with the highest number of channels with an interference above average is chosen to be assigned a channel next. If there is more than one node with the same maximal number of interference values above average, the node with smaller id is chosen. A schematic representation of this process can be seen in Fig. 3.2

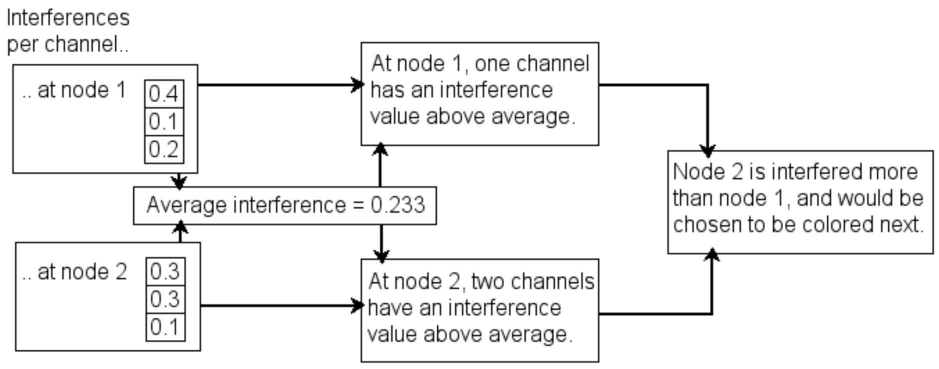

Interferences per channel..
.. at node 1
0.4
0.1
0.2
.. at node 2
0.3
0.3
0.1
Average interference = 0.233
At node 1, one channel has an interference value above average.
At node 2, two channels have an interference value above average.
Node 2 is interfered more than node 1, and would be chosen to be colored next.

Figure 3.2: Schematic representation of the process to determine the next node to be colored in Greedy Most Interfered First in an example with two nodes.

Greedy Most Interfered First ensures, that nodes with a restricted choice of channels are assigned a channel as early as possible.

3.3.3 Tabu Greedy (Breadth First)

Based on the fact, that there is a high probability, of the last link on the route from a router to the gateway to be a bottleneck, we developed the following slight modification of Greedy Breadth First.

We allocate one exclusive channel to each gateway and assign the remaining channels in Greedy Breadth First manner to the remaining nodes.

Note, that for reasonable applying this algorithm, we need to ensure that the number of gateways is smaller than the number of channels.

3.4 Channel Allocation based on the Merge Algorithm

The Merge algorithm is also iterative like the greedy algorithms, but Merge works with another strategy. Merge starts by allocating an unique channel to each node. Then it iteratively merges two channels into one until it reaches the desired number of channels.

After assigning a virtual channel to each node, for each pair of channels the algorithm iteratively computes the virtual interference values between all nodes assigned to one of these channels. In the first step there are just single nodes with unique channels and the algorithm checks the pairwise interferences. Based on these computation Merge then choses the two channels that interfere the least with each other and merges them into one channel. This is done as long as we didn't reach the actual number of channels available. (see also Fig. 3.3)

Merge uses a merge table ($MT[\#channels][\#nodes]$) to represent the network with its connections. In each table cell of MT we store three values

- a boolean value if the nodes base station interface is tuned to this channel
- a boolean value if the nodes subscriber station interface is tuned to this channel
- the virtual interference value a node senses on this channel

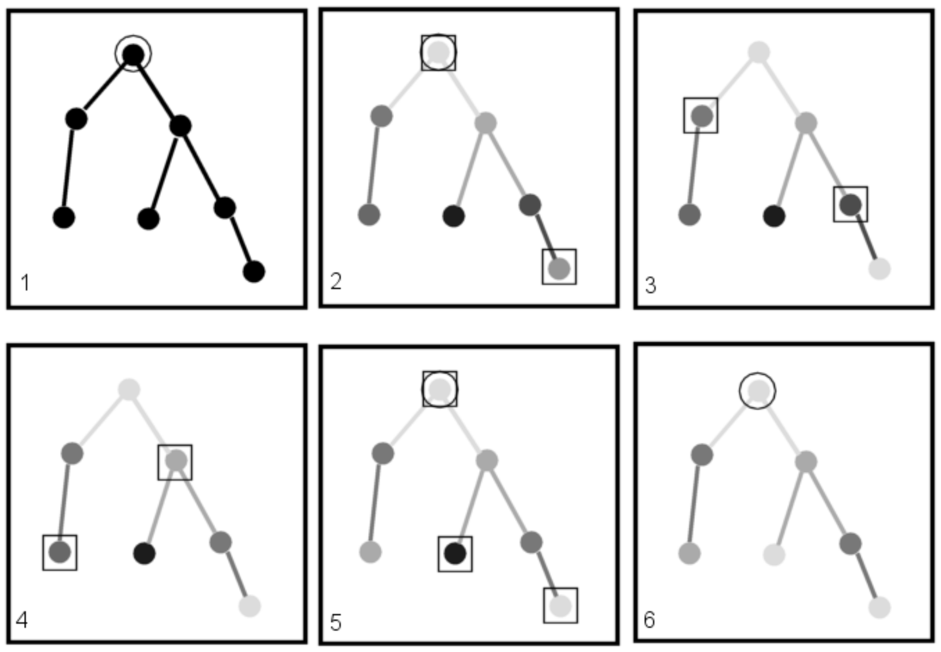


Figure 3.3: Sequence of the Merge algorithm in an example network with 6 routers and 1 gateway (circled). The nodes belonging to one of the channels to be merged in the next step are marked with a square.

In the beginning the number of channels is equal to the number of nodes, then iteratively, two rows are merged into one.

The initial settings in MT are the following:

- In each cell $MT[c_i][n_i]$ the value for the base station interface is set to 1. It is set to 0 for all other cells $MT[c_i][n_j]$ with $i \neq j$.
- The subscriber station value in $MT[c_i][n_j]$ is set to 1 if there is a directed edge from node n_j to node n_i and to 0 else.
- The interference value of cell $MT[c_i][n_j]$ is the virtual interference value node n_j senses on channel c_i.

To decide, what two rows we should merge, when we have multiple options, we compute the interference on the assignment for every possible merge operation of two rows (channels). For every node having an interface tuned to any of the two channels we are considering, we sum up both of the interference values. Then the two channels with the minimal interference are merged.

To merge two rows (channels) i and j the following rules hold:

- If for any of the columns (nodes) more than one interface is tuned to one of the two channels, the merge of these two rows is not possible.
- If the base station interface value of a node in any of the merged rows is set to 1, the base station interface value of the new row is also set to 1 else it is set to 0.
- If the subscriber station interface value of a node in any of the merged rows is set to 1, the subscriber station interface value of the new row is also set to 1 else it is set to 0.

- The interference value of the new row is set to the sum of the interference values of the merged rows.

3.4.1 Example

As a simple example we look at a network of four nodes that form a chain with equal distance between the nodes (Fig. 3.4).

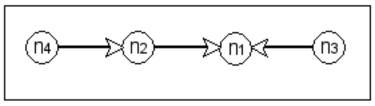


Figure 3.4: Example network with four nodes. An arrow signifies "subscribed to".

In Tab. 3.1 we see the example interference values between two interfaces according to the hop distance of the interfaces or the corresponding nodes respectively.

hop distance	interference
0	2.0
1	1.0
2	0.5
3	0.2

Table 3.1: Abstract interference values per hop distance for the Merge example

With these information we can now build the initial Merge Table we see in Tab. 3.2.

	n_1	n_2	n_3	n_4
c_1	(1, 0, 2.0)	(0, 1, 2.0)	(0, 1, 2.0)	(0, 0, 0.5)
c_2	(0, 0, 1.0)	(1, 0, 2.0)	(0, 0, 0.5)	(0, 1, 2.0)
c_3	(0, 0, 1.0)	(0, 0, 0.5)	(1, 0, 2.0)	(0, 0, 0.2)
c_4	(0, 0, 0.5)	(0, 0, 1.0)	(0, 0, 0.2)	(1, 0, 2.0)

Table 3.2: Initial Merge Table of the example network.

We can see, it's not possible to merge c_1 and c_2 as n_2 has an interface on each of the channels. Also it's not possible to merge c_1 and c_3 because of n_3 or c_2 and c_4 because of n_4. Remain the possible merge operations listed in Tab. 3.3 along with their computed interference values.

As c_3 and c_4 have the smallest interference sum, the algorithm now merges these two channels into c_3 according to the rules defined above. The resulting merge table can be seen in Tab. 3.4

3.4.2 Implementation in the Field

The Merge algorithm is hard to distribute, but we evaluated it nevertheless to compare the results to easy to distribute and comparatively simple greedy algorithms.

In the beginning we start with an unlimited number virtual channels, but we can't get real interference values for virtual channels. But if this algorithm turns out to be outstanding, it would be possible to implement it in a field test by executing a precomputation, where each antenna is the only one to send a signal for some time, and all antennae record the signal strength of each other antenna. With this data, they can also compute the virtual interferences of the virtual channels.

channel A	channel B	interference value
c_1	c_4	$(2.0 + 0.5) + (2.0 + 1.0) + (2.0 + 0.2) + (0.5 + 2.0)$ $= 10.2$
c_2	c_3	$(2.0 + 0.5) + (0.5 + 2.0) + (2.0 + 0.2) = 7.7$
c_3	c_4	$(2.0 + 0.2) + (0.2 + 2.0) = 4.4$

Table 3.3: Possible pairs of channels to merge along with their interference sum.

	n_1	n_2	n_3	n_4
c_1	(1, 0, 2.0)	(0, 1, 2.0)	(0, 1, 2.0)	(0, 0, 0.5)
c_2	(0, 0, 1.0)	(1, 0, 2.0)	(0, 0, 0.5)	(0, 1, 2.0)
c_3	(0, 0, 1.5)	(0, 0, 1.5)	(1, 0, 2.2)	(1, 0, 2.2)

Table 3.4: Merge Table of example network after merging c_3 and c_4 into c_3

Chapter 4

Evaluation

This chapter shows the results of the evaluation of the topology construction algorithm and the channel allocation algorithms.

4.1 Evaluated Scenarios

To evaluate the algorithms we created different scenarios with nodes placed randomly and nodes placed on a grid with equal distance to each other.

To create these scenarios, we had to consider the maximum capacity of a node and the communication range. The communication range of the presented nodes is about 115m. Empiric test showed that with three nodes per $1000m^2$ there is a high probability for a connected graph. The maximum capacity of an node is assumed to be 54Mbps. Also assuming, each router needs a capacity of 2Mbps to a gateway and also from a gateway to itself, this leads to a maximum of 13 nodes connected to one gateway. Taking into account possible interferences, we will place about one gateway per 11 nodes.

We evaluated eight different scenario classes, four of them with random placed nodes and four of them with the nodes arranged in a grid pattern with a distance of 100m between the nodes. (Tab. 4.1)

Scenario class	Placement	#Routers	#Gateways	Area
Rand12	random	11	1	200m x 200m
Rand24	random	22	2	300m x 300m
Rand36	random	33	3	400m x 300m
Rand48	random	44	4	400m x 400m
Grid25	grid	23	2	400m x 400m
Grid36	grid	33	3	500m x 500m
Grid49	grid	45	4	600m x 600m
Grid64	grid	58	6	700m x 700m

Table 4.1: Overview of the considered scenarios with the category of placement, the number of routers, the number of gateways and the area on which the nodes are placed.

For each of the scenario classes we created five different instances. In random placement scenarios for each instance of the scenario, all the nodes are replaced randomly. In grid placement scenarios the positions of the nodes are predetermined, so only the positions of the gateways are randomly variated.

4.2 Topology Construction Results

To evaluate the topologies, we calculate the minimal capacity a node can reach, only bounded by the topology and the fairness index of the distribution of nodes per gateway after Jain's Fairness Index [9]. In our implementation, we are able to pass the topology construction algorithm a hard upper bound for the number of routers per gateway, and if the constructed topology has any gateway with more routers connected to it, the topology algorithm returns a failed-message. To evaluate the performance of the topology construction algorithm, the upper bound was set to the total number of nodes.

4.2.1 Capacity

In Fig. 4.1 we can see the optimum minimal capacities that can maximally be reached for every class of scenarios along with the minimal capacities of the constructed topologies. To compute the optimum of a scenario, we assume the nodes are equally distributed on the gateways and divide the number of nodes by the number of gateways. Then the link capacity is divided by this ratio to determine the maximal minimal capacity.

The minimal capacities of the constructed topologies are based on the maximal number of nodes connected to one gateway. The link capacity is then divided by this maximal number of nodes to determine the minimal capacity of this topology.

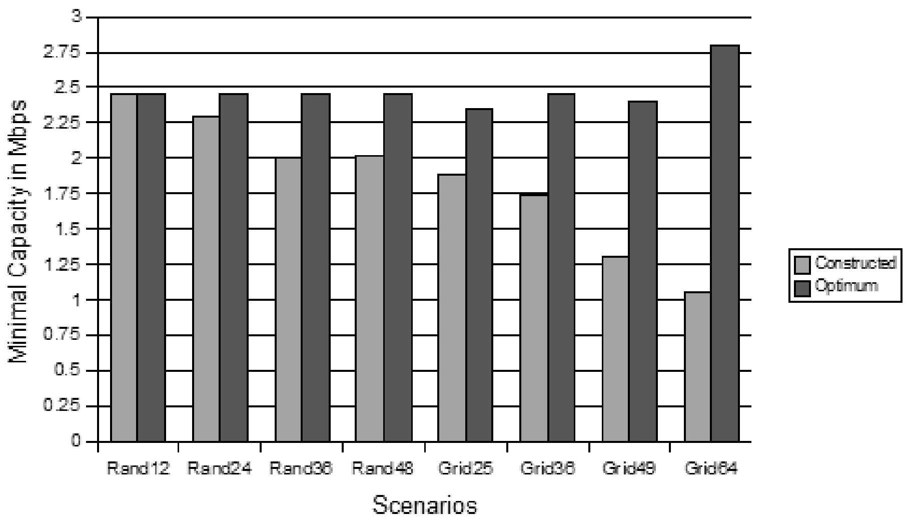


Figure 4.1: For each scenario class, we see the average minimal capacity of topologies constructed on the five instances of the scenarios (Constructed) and the minimal capacity for an optimal topology in the given scenario class (Optimum).

4.2.2 Fairness

An optimally constructed topology, with an equal number of nodes connected to each gateway, has a fairness index of 1. Looking at the fairness indices of the distribution of nodes to gateways, we see an average fairness index of 0.869 with a standard deviation of 0.179.

The minimum fairness value is 0.389. It is reached in a Grid64 scenario topology, where we have the following distribution of nodes per gateway:

```
{5, 39, 14, 1, 3, 2}
```

Looking at the node respectively gateway placement of this scenario instance, we see that 4 of the 6 gateways are neighbors, while the other 2 gateways also neighbors. The bad topology is not inevitable, by hand we can easily construct an optimal topology, but obviously the algorithm has a problem with this constellation. We assume the problem lies in the balancing heuristic. We will discuss this problem in Chapter 6.

4.3 Channel Allocation Results

For evaluating the channel allocations, we computed capacity, interference and fairness values as explained in Chapter 2.

The following results are all obtained by letting the algorithms allocate 12 channels to the nodes. Results for tests with only 3 channels can be found in Appendix A.

4.3.1 Capacity

Overall View

In Fig. 4.2 and Table 4.2 we see the average values over all scenario instances for the average capacity of all routers and the average minimum capacity of any router. Along with the average upper bound of the minimum capacity of all the instances.

We see that for both capacity criteria, the three greedy algorithms have about the same average value. For the average of the minimal capacities the Greedy Breadth First algorithm has the best value, but only 0.007 Mbps better than the Tabu Greedy algorithm. The Tabu Greedy algorithm reaches the highest overall average capacity with 2.348 Mbps, which is 0.033 Mbps more than the Greedy Breadth First algorithm.

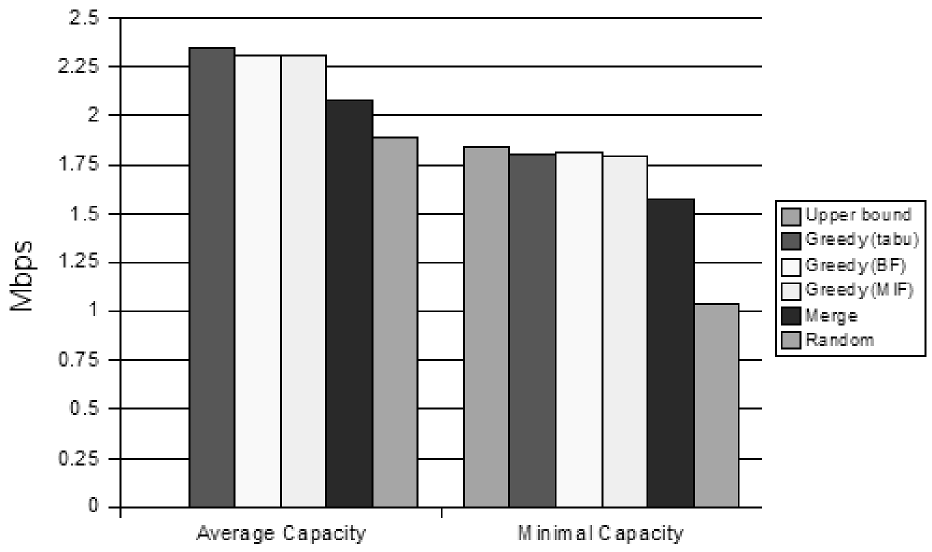


Figure 4.2: Average values for the average and the minimal capacities over all scenario instances grouped by algorithm. Along with the upper bound for minimal capacity determined by the topology construction.

Algorithm	Avg Cap in Mbps	Min Cap in Mbps	in % of u.b.
Upper Bound	n.a.	1.843	100%
Greedy (Tabu)	2.348	1.808	98%
Greedy (BF)	2.315	1.815	98%
Greedy (MIF)	2.308	1.796	97%
Merge	2.078	1.575	85%
Random	1.894	1.035	56%

Table 4.2: Average numeric values for the average and the minimal capacities over all scenario instances grouped by algorithm.

Minimum Capacities

We will now have a detailed look at the minimum capacities that result from the channel allocation algorithms for the different scenarios.

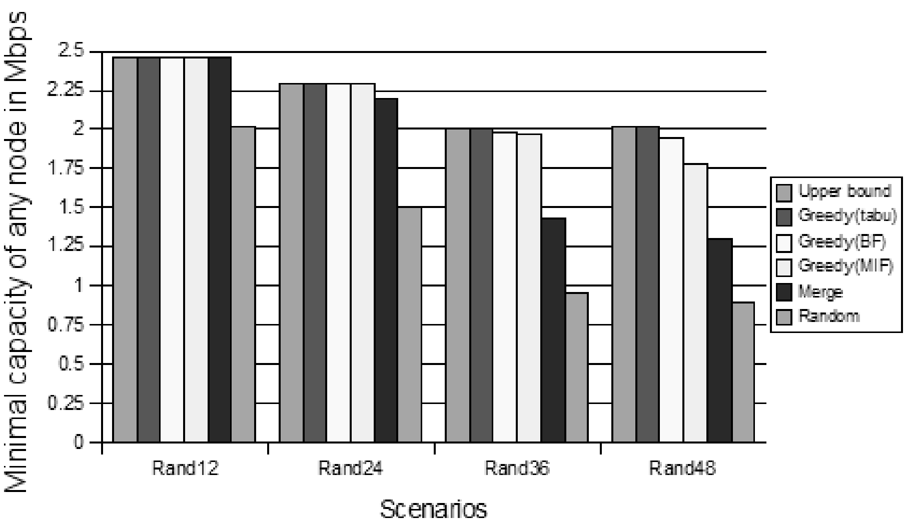


Figure 4.3: Average Minimum Capacities, over all instances of a scenario, of assignments in random placement scenarios

Fig. 4.3 shows the average results of the minimal capacity for the random placement scenarios. Again, each value is the average value of five scenario instances. For Rand12 every algorithm reaches the, not so difficult to reach, upper bound. For Rand24, the greedy algorithms all still reach the upper bound value, while the Merge algorithm is only close to the upper bound. In Rand36 and Rand48 we can see a tendency of the greedy algorithms differing. The Tabu Greedy algorithm reaches the upper bound for all the random placement scenarios. The other greedy algorithms are close to the upper bound for Grid36. For Rand48, Greedy Breadth First is still quite good, while the Greedy Most Interfered First algorithm only reaches about 88% of the upper bound. Looking at the Merge algorithm in Grid36 and Rand48, it only reaches 71% and 65% of the upper bound.

Fig. 4.4 shows the average minimum results for the grid placement scenarios. This figure shows a slightly different situation as the figure for the random placement scenarios.

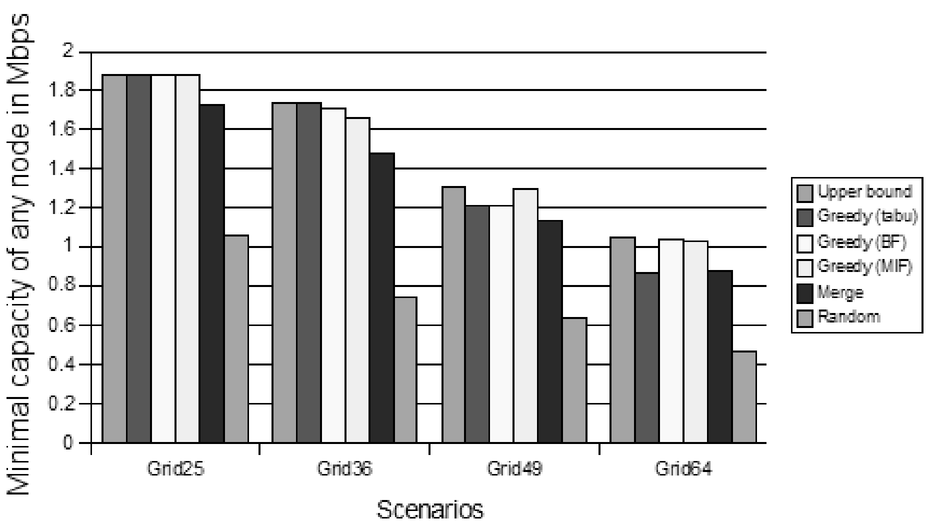


Figure 4.4: Average Minimum Capacities, over all instances of a scenario, of assignments in grid placement scenarios

For Grid25 and Grid36, the results of minimal capacities look still quite the same as the results of Rand24 and Rand36 (Fig. 4.3). But for Grid49, the Greedy Most Interfered First almost reaches the upper bound, while the Tabu Greedy algorithm only reaches about 93% of the upper bound. For Grid64 also the simple Greedy Breadth First comes close to the upper bound and even the Merge algorithm has a slightly better result as Tabu Greedy, which only reaches 82% of the upper bound.

A possible explanation for this phenomenon could be that in large grids, the paths in hops to the gateways are much longer than in random placements. Therefore also the router one hop from a gateway has a high traffic load. In Tabu Greedy assignments, the channels assigned to gateways are protected from interferences, but not the channels of the routers one hop from gateways. Of course also in the other assignments these channels are not treated specially, but unlike the Tabu Greedy algorithm, other algorithms assign a higher number of channels to all the routers. This produces lower interference values, because the probability of another router having the same channel assigned is lower.

It would be interesting to investigate, where the bottleneck links are in these assignments.

Average Capacities

Now we have a detailed look at the average capacities that are produced by the channel allocation algorithms.

Fig. 4.5 shows the average capacities of all nodes in assignments to random placement scenarios. For Rand12 and Rand24, the average capacities for all algorithms are relatively high. For Rand36 and Rand48 we see some differences. The Merge algorithm reaches a apparent lower average than the greedy algorithms. Within the greedy algorithms, we see Tabu Greedy always with the maximum average, while the other two greedy algorithms alternate on the runner-up.

In Fig. 4.6 we see the average capacity values of the assignments computed on grid placement scenarios. For the scenarios from Grid25 to Grid49 we basically

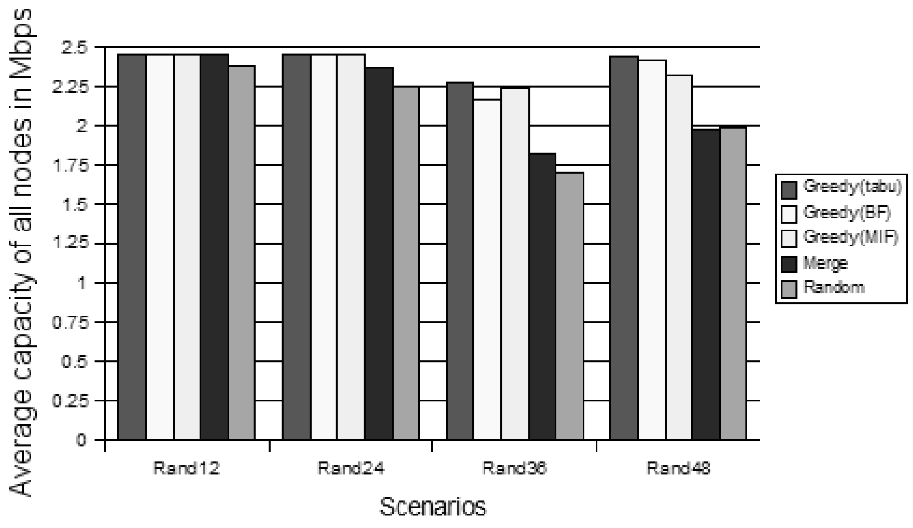


Figure 4.5: Average Capacities of assignments of random placement scenarios ordered by scenario

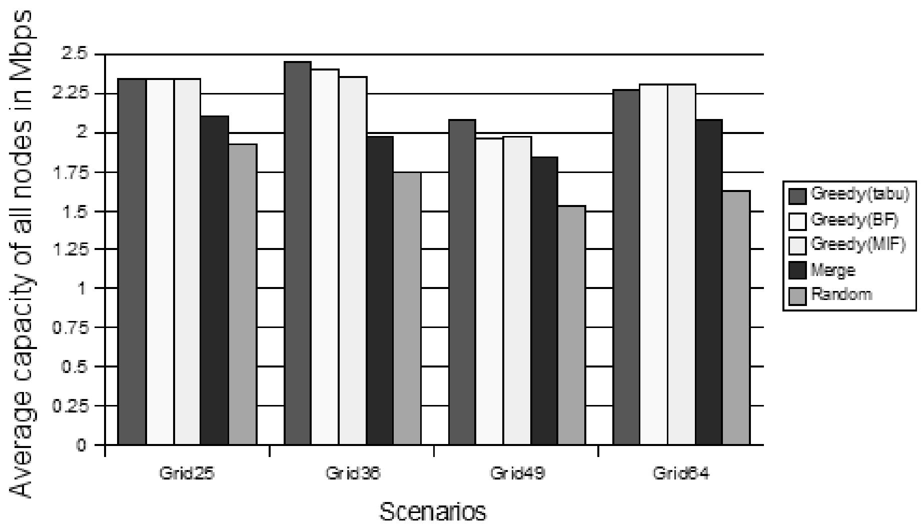


Figure 4.6: Average Capacities of assignments of grid placement scenarios ordered by scenario

have the same picture as for the random placement scenarios. The Tabu Greedy on top, Merge behind, and the other two greedy algorithms in between. For Grid64, as at the minimal capacity figure (Fig. 4.4), we see Tabu Greedy falling back, while Greedy Breadth First and Greedy Most Interfered First share the top rank.

4.3.2 Fairness

As a second evaluation criteria, we look at the distributions of channels on the network. For this we calculated the fairness indices of the distributions of traffic flows on channels, of base station interfaces on channels and of links on channels.

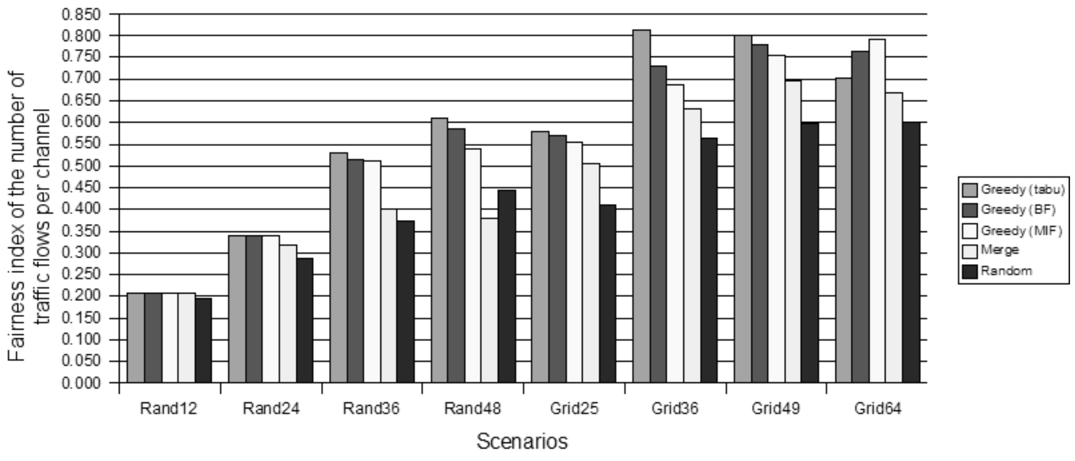


Figure 4.7: Fairness indices of the distribution of traffic flows on channels of each algorithm ordered by scenario

Fig. 4.7 shows the fairness indices of the distribution of traffic flows on channels. Meaning, how many traffic flows are on each channel. We see that the fairness for scenarios with a small number of nodes are quite low. This is, because the number of traffic flows on channels at gateways is too high to be compensated by traffic flows on other links. And as we see at the example of Rand12, a low fairness index doesn't have to imply a bad channel allocation.

Fig. 4.8 shows the fairness indices of the distribution of base station interfaces on channels. This is, how many nodes are accepting subscribing nodes on each channel. Here we see that Tabu Greedy gets worse with ascending number of nodes. Looking at the example of Grid64 we can easily see, there are six channels only used by one node (the gateways) and six channels used by a total of 58 nodes, that makes an average of 9.66 nodes per channel. The other algorithms have 12 channels for 64 nodes, which leads to a ratio of only 5.33 nodes per channel. This results in the striking disequilibrium reflected in the fairness index.

Fig. 4.9 shows the fairness indices of the distribution of links on channels. Note that the number of links per channel is equal to the number of subscriber station interfaces per channel. The correlation between the Link fairness and the Traffic Flow fairness is 0.97. This is surprising because actually there are big differences between the links concerning the traffic flows on them.

4.3.3 Interferences

As third evaluation criterion we look at the interference values. Interferences are indicated as attenuation of the transmitted signal strength, which is assumed to be

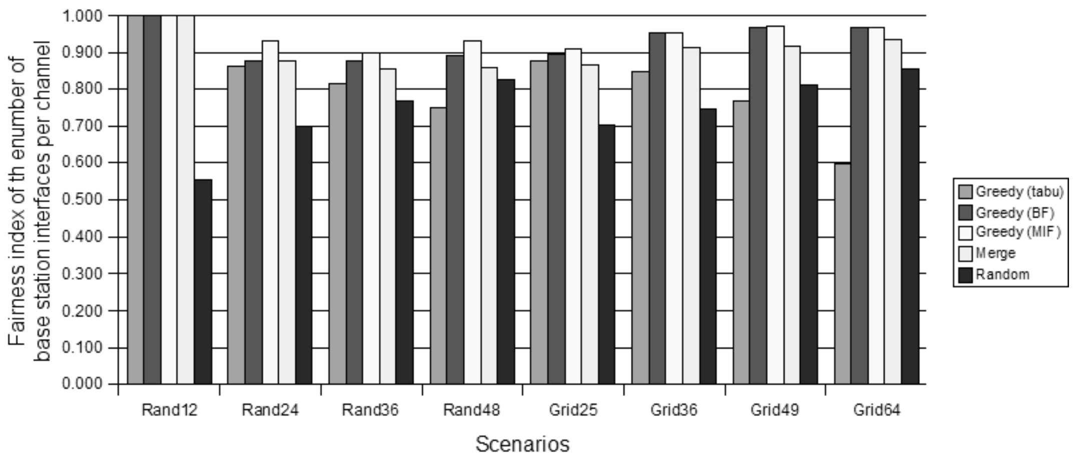


Figure 4.8: Fairness indices of the distribution of base station interfaces on channels of each algorithm ordered by scenario

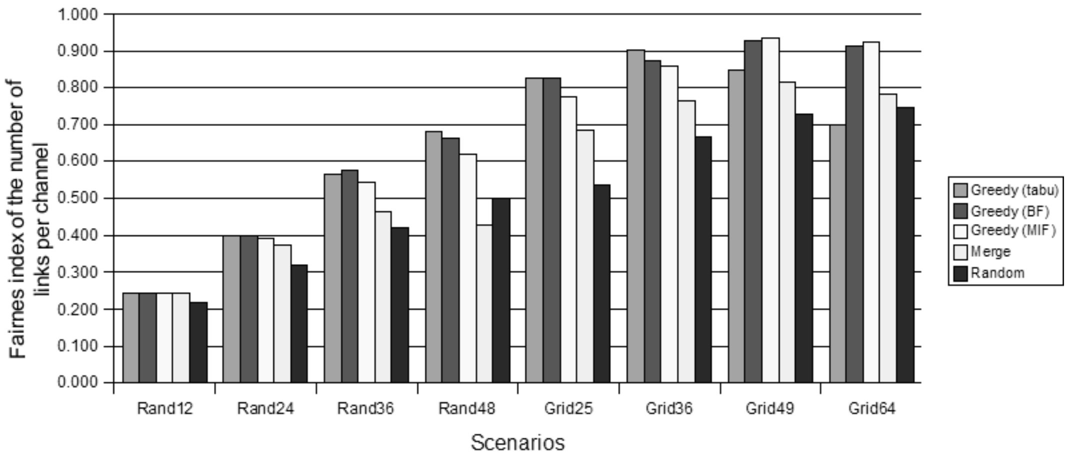


Figure 4.9: Fairness indices of the distribution of links on channels of each algorithm ordered by scenario

the same for all nodes.

Overall View

In Fig. 4.10 we see an overview of the average interferences and the average maximal interferences for each algorithm. The averages are computed over all scenario instances evaluated.

For the average interferences and for the average maximal interferences, we see the same relative picture. Random has the highest interference values, about a factor in the dimension of 1000 times higher than the values of the other algorithms. Among the other algorithms, Tabu Greedy produces the highest interferences, followed by Merge, Greedy Most Interfered First and Greedy Breadth First.

We expected Tabu Greedy to produce high interference values, as the routers share a smaller number of channels. A little surprising is the fact that Greedy Breadth First has lower interference values than Greedy Most Interfered First, as the latter always assigns a channel to the most restricted node next which we expected to lead to less interference.

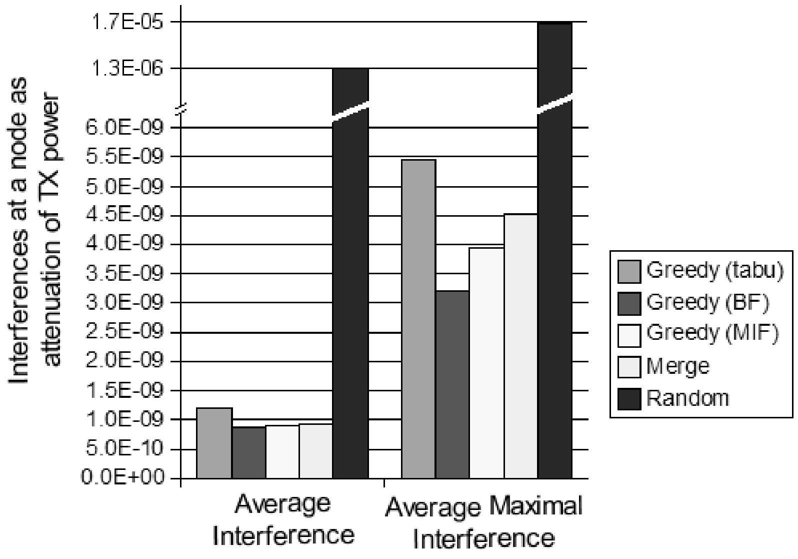


Figure 4.10: Average values over all scenario classes for the average interference and the average maximal interference grouped by algorithms

Maximum and Average Interferences in Detail

We now have a more detailed look at the results of the interference evaluation. In Fig. 4.11 we see the maximal interferences of any instance of a scenario class for each algorithm. In Fig. 4.12 we have the average interferences over all instances of a scenario class for each algorithm.

The detailed views reflect the overall view of the relations between the algorithms, with ascending interferences on ascending number of nodes.

The values of the maximal and the average interferences look quite similar, and they also have a high correlation of 0.994

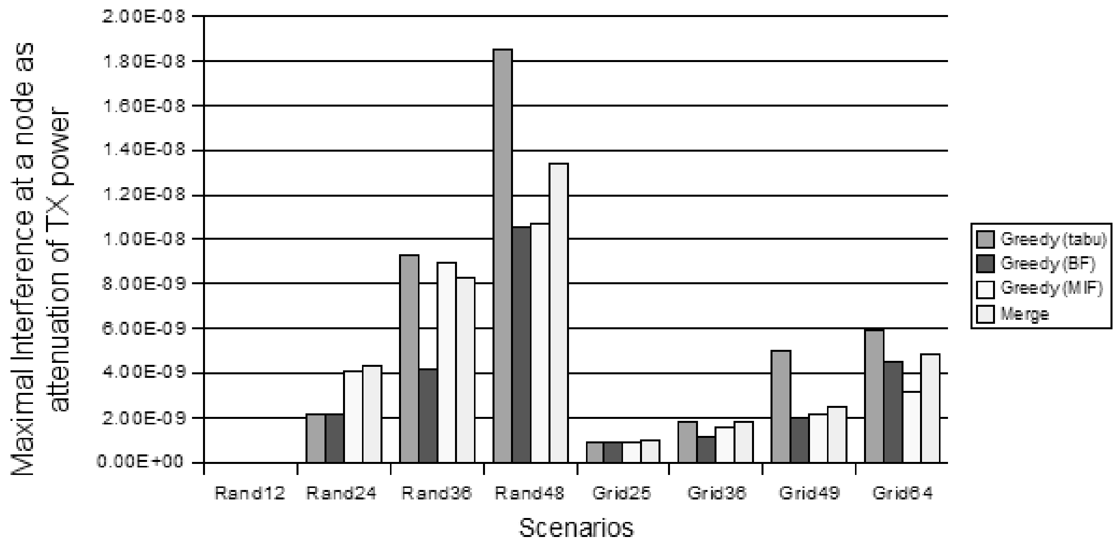


Figure 4.11: Maximal interferences of any instance of a scenario class for each algorithm

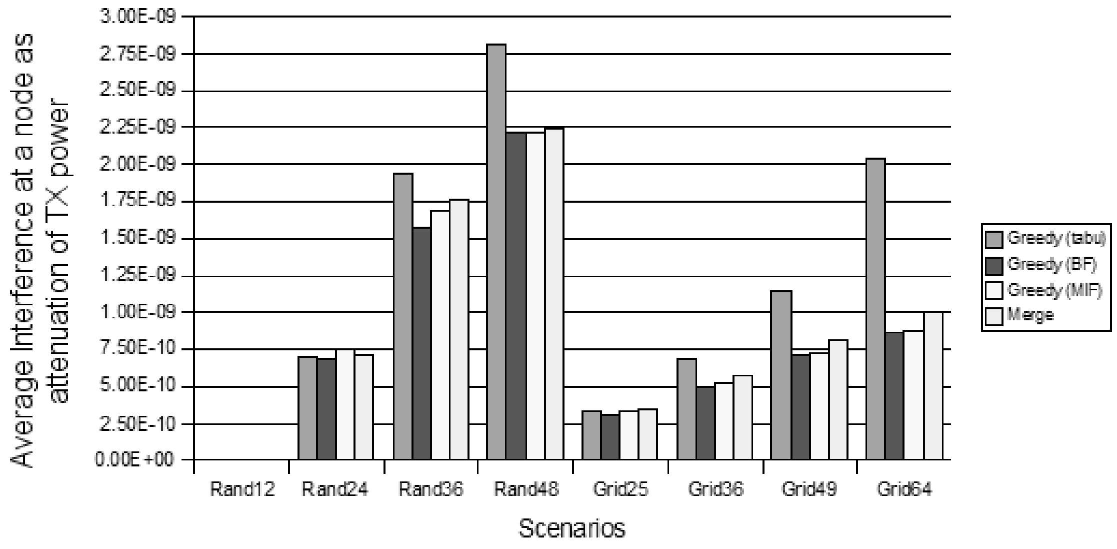


Figure 4.12: Average interferences over all instances of a scenario class for each algorithm

4.4 Joint Results

4.4.1 Correlation between topology fairness and channel allocation results

As we saw, the constructed topology is related to the upper bound for minimal capacities as also on the minimal capacity results from the allocation algorithms. Therefore we computed the correlations between the fairness indices of the constructed topologies and the average as well as the minimal capacities of the algorithms over all instances of scenarios.

Correlations with fairness	Average Capacity	Minimal Capacity
Upper Bound	n.a.	0.94
Greedy (Tabu)	0.71	0.60
Greedy (BF)	0.61	0.84
Greedy (MIF)	0.62	0.52
Merge	0.42	0.30
Random	0.79	0.55

Table 4.3: Correlation of the fairness value of constructed topologies with the average capacities and the average minimal capacities reached by the different algorithms.

In Table 4.3 we see that the upper bound for the minimum capacity correlates highly positive with the fairness value. But also the minimum and average capacities reached by the algorithms correlate positively with the fairness of the topologies.

This means the fairer the topology is constructed the better are the results of the algorithms.

Chapter 5

Related Work

This chapter shows research that have been done in the areas of channel allocation and graph coloring.

5.1 Channel Allocation

5.1.1 Channel allocation for a given topology

In [8] Tang et al elaborate a channel assignment algorithm for channel assignment for a given network. The algorithm works with potential interferences, and "“hopes”' to achieve a low co-channel interference this way. They assume traffic demands are not given and potentially fluctuating over time. As resulting assignment, they guarantee a k-connected network, a main feature of this work.

Das [14] uses two mixed integer linear programming models for solving the fixed channel assignment problem with multiple radios optimally, but traffic independent. Das considers an 802.11-based network and thus symmetric links, he also assumes traffic between any nodes.
The optimality criterion they use is maximization of the number of possible simultaneous transmissions in the network. They maximize the 1-hop throughput as opposed to multi hop end-to-end throughput, because maximum network layer throughput depends on the real traffic, which they expect to be fluctuating.

Kodialam [15] examines wireless mesh networks assuming fast channel switching is possible. Based on this, he developed a dynamic Channel Allocation algorithm, but also a fixed channel allocation algorithm, Balanced Static Channel Assignment (BSCA), which is almost greedy. Kodialam also utilize graph coloring, but instead of nodes, he colors the edges of the graph.

5.1.2 Joint Topology Construction and Channel Allocation

Although in this work, we worked on channel assignment for a given topology, we found some interesting work for the joint topology construction and channel allocation problem.

Raniwala et al. ([3], [16] and [2]) present an algorithm that utilize only local topology and traffic load information to assign channels. The load aware channel assignment is used to automatically form a fat-tree where more bandwidth is available on links closer to the roots of the trees, i.e. gateways.

Nodes also have multiple interfaces, divided into UP-NICs (to communicate with its parent node) and DOWN-NICs (to communicate with its children nodes) which correspond to our subscriber and base station interfaces. Each node is responsible for the assignment of channels to its DOWN-NICs. To assign a channel to a DOWN-NIC, a node needs to estimate the usage status of all the channels within its interference neighborhood, the total load of a channel. As higher nodes in the spanning tree (nearer to the gateway) need more bandwidth, they are given a higher priority in channel assignment. When a node performs channel assignment, it restricts its search to those channels that are not used by any of its interfering neighbors with a higher priority.

Because traffic pattern and thus channel loads can evolve over time, the channel assignment is adjusted periodically in a channel load-balancing phase, where each node evaluates its current assignment with respect to the actual channel loads. To deal with node failures, each node remembers alternative ADVERTISE messages, it has received from all other potential parent nodes. Upon detection of a parent-node failure, each of its child nodes joins a "'backup"' parent node to re-establish connectivity to a gateway.

A work by Alicherry et al. [4] attempts to solve the throughput maximization problem for a MultiChannel-MultiRadio network, where the network is restricted to be a superset of a disk graph, i.e., the interference range is assumed to be a fixed multiple of the communication range. They mathematically formulate the joint channel assignment and routing problem and try to solve it using linear programming. The algorithm works in multiple rounds, first the LP problem is solved, then the channels are assigned. After that there is a Post Processing round, a Flow scaling round and in the end, they try to make the assignment interference free.

So [17] uses a similar architecture as we do, but with only one NIC (network interface card) per node. There are also multiple gateways that offer connectivity to the wired network. The resulting architecture are multiple trees, with all nodes of the trees working on the same channel, specified by the according gateway.

Wei et al. [18] developed an interference-aware IEEE 802.16 framework with a design goal of achieving overall high utilization of the WiMax Mesh network. Their proposed scheme includes a novel interference-aware route construction algorithm and an enhanced centralized mesh scheduling scheme, which consider both traffic load demand and interference conditions.

5.1.3 Other principles

Switching Radios Kyasanur [19] builds multiple radio networks with IEEE 802.11 network interface cards. One of the radios remains fixed on one channel, while the other radios are switching between the other channels to communicate with the fixed channel radios of neighboring nodes.

Base Channel Assignment Marina [13] also works with a 802.11-based multi-radio wireless mesh network. The main idea is to assign channels to radio interfaces to obtain an initial, well-connected topology in a traffic-independent manner such that a pair of neighboring nodes have a common default channel to communicate with each other, which we refer to as the base channel assignment. When channel switching delays are large, base assignment itself can be used for all communication. But they expect switching delays to reduce and making dynamic channel assignment feasible.

In their greedy heuristic channel assignment algorithm *Connected Low Interference Channel Assignment* (CLICA) they try to minimize an overall measure of interference. If coloring a node has a serious impact on the flexibility of a neighboring node to chose a color, they color this node next.

Directional Antennae Raman [20] connected indian villages with a mesh network out of directional IEEE 802.11 (WiFi) antennas. They use the recently designed 2P MAC protocol ([21] and [22]). They explore several heuristics for channel allocation and find a set of heuristics that achieve the optimal allocation in most scenarios. These heuristics also include greedy coloring.

Genetic Algorithm Vanhatupa et al. [23] just recently developed a new version of a genetic Algorithm for channel allocation.

5.2 Graph Coloring Algorithms

Given the network topology, graph coloring algorithms (see [24] or [5]) can be applied for channel allocation relatively easy. In this section, we will look at some graph coloring algorithms, adaptations to handle interferences are already made in Chapter 3.

5.2.1 Greedy Coloring

The Greedy algorithm, also known as the sequential algorithm, has several variations elaborated on Joseph Culberson's Graph Coloring Resources Page [25].

The Greedy algorithm takes each vertex in turn in some predefined order and tries to color the vertex with one of the colors used so far. In other words, it tries to add the vertex to one of the existing color classes. If it is not possible to color it with any existing color, then a new color class is created and the vertex is assigned the color of that class.

This leaves two obvious choices:

1. If there is more than one existing color class the vertex can enter, which one should be chosen?
   - ”Simple Greedy” visits the classes in order 1..k, and is the technique usually indicated when someone refers to the sequential algorithm.
   - ”Largest First Greedy” ranks the classes by number of vertices currently in them, and searches by descending size. This should tend to build larger independent sets.
   - ”Smallest First Greedy” ranks the classes by number of vertices currently in them, and searches by ascending size. This should tend to equalize the size of the color classes.
   - ”Random Sequence Greedy” searches the sets in a random order.
   - ”Reverse Order Greedy” searches the sets in the order k..1.
2. What preordering to assign to the vertices?
   - inorder (presents the vertices in the order 1..n)
   - random
   - decreasing degree
   - increasing degree

5.2.2 Backtracking

The first coloring algorithm of this type by R.J. Brown [26] is based on the following simple backtracking rule. Suppose that the vertices of a graph G have been ordered in some way and are reindexed so that V_i is the i^{th} vertex in this ordering. Then, for a given set of colors, two steps: forward and backward, are performed alternately.

Forward: The forward step colors the vertices sequentially up to a vertex which cannot be colored because of a lack of colors available to it.

Backward: The backward step moves sequentially back in search of the first vertex which can be colored with another feasible color and then the forward step is resumed, etc.

Backtrack: If a new, better coloring of G has been found, the algorithm attempts to find the next one. The activity terminates when a backtrack reaches V_1.

Kubale and Jackowski [27] made a survey of different backtracking algorithms and also provide an algorithmic framework for backtracking i.e. implicit enumeration algorithms, based on the backtracking algorithm by Brown [26].

5.2.3 Merge Coloring

Juhos et al. ([28] or [29]) present a new model for solving the graph k-coloring problem. To start, they color each vertex of the graph with an unique color. Then they iteratively merge two colors into one, until there is no possibility left to merge two colors without violating the constraints of graph coloring.

The colored graph is represented in a $k \times n$ table called Merge Table (MT). Where k is the number of colors and n is the number of vertices. One of three possible values must be assigned to every cell of the table: 1, 0 or X.

1: If a vertex i is colored with color c then set cell $MT(c, i)$ to 1.

0: A cell $MT(c, i)$ is set to 0, if the color c assigned to i would violate a constraint, i.e. an edge exists between this vertex and another vertex colored with c.

X: If it doesn't matter whether we color a vertex i with color c, we set cell $MT(c, i)$ to X.

MT is initialized as $n \times n$ matrix, assigning color i to vertex i.

- $MT(i, i) = 1$ for every i
- For every edge (i, j) in the graph we set cell $MT(i, j) = 0$
- All the remaining cells of MT are set to X.

By merging rows, we then can reduce the number of colors used. The only situation, two rows can't be merged, is if at least one column exists where one cell is set to 0 and in the other row the cell set to 1. The values of a merged row based on the values of the two rows to be merged are shown in Tab. 5.1.

Merges are executed as long as possible, or until a desired number of colors k is reached.

Note that not all values for k are reachable and the reachable k is determined by the order of the merges.

Column in row A	+	Column in row B	=	Column in merged row
1		X		1
1		1		1
0		X		0
0		0		0
X		X		X
1		0		row merge is canceled

Table 5.1: Resulting values after merging two rows into a new row. For each column considering the value in row A and in row B, the value for the merged row can be read in this table.

Chapter 6

Discussion and Future Work

In this thesis, we studied different channel allocation algorithms and a topology construction algorithm. To evaluate them, we created different node placements with a varying number of routers and gateways.

On the node placements we constructed topologies. We evaluated these topologies by comparing the minimal capacity of any router with the capacity every router would get if the routers were distributed equally over all gateways.

On the topologies we assigned channels to all base station interfaces and therewith also to the subscriber station interfaces and links between them. For these channel allocation schemes we again calculated the capacity values, considering also interferences, for every router. We compared the average and the minimal capacities of the different algorithms with each other and the minimal capacity additionally with the upper bound given by the topology. Further we also computed the fairness values of the distribution of channels on traffic flows, base station interfaces or links. And also the interference values each node has on the channel its base station interface is tuned to.

We will now have a look at the major findings of this thesis, their meaning and possible improvements.

6.1 Channel allocation

The main evaluation criteria is the minimal capacity assigned to a node, as in our context of the TOWN project, each node has a fixed traffic demand. If a node gets a potential capacity which is higher than its demand, the difference just remains unused. But if the capacity of a node is lower than its demand, this is a major constraint to the telephony application.

Therefore, if we look at the overall view of the capacities (Fig. 4.2) we see that the results of the greedy algorithms are at about the same level, while the merge algorithm performs slightly worse, concerning the minimal capacity as also the average capacity. Also if we look at the single scenario results (Fig. 4.3 and Fig. 4.5) for the minimal capacity, we see that the merge algorithm almost never outperforms the greedy algorithms. The only exception is the performance of Tabu Greedy on large grids of 49 or 64 nodes. Thus the first conclusion we can draw, the Merge algorithm does not perform as well as the greedy algorithms.

Comparing the greedy algorithms with each other, the results are not that clear. For small random placement networks, up to 24 nodes, all greedy algorithms always reach the upper bound. For random placement scenarios, the Tabu Greedy algorithm performs a little better than the other greedy algorithms. But we expect the Tabu Greedy algorithm to become worse relative to the other greedy algorithms with increasing number of nodes. Because the interferences between non-gateway nodes will have a bigger negative impact on the capacities, than the protection of the gateway channels is an improvement. The other greedy algorithms should perform better, because they have a bigger choice of channels, as no channels are allocated exclusively to gateways.

6.1.1 Improvements

Although the greedy algorithms already perform quite well, there is potential for improvements. Instead of allocating channels only based on the interference values, we can also take into account the traffic loads of each node. E.g. multiplying the interference by the number of traffic flows a node sends and receives. For networks with a small number of nodes, this would result in similar assignments as the Tabu Greedy algorithm produces, as gateways still have a relatively high traffic load. But with increasing number of nodes, the relative traffic load of the channels of gateways decreases and there are situations where we better reuse the channel assigned to a gateway.

The Most Interfered First algorithm might be improved by using another metric than the number of interference values above average to chose the next node. For example the maximal sum of all interference values below average or the node with the maximum of minimal interference values of all nodes. It's not obvious that this improves the algorithm, it's just a proposal for other metrics that need to be evaluated.

6.2 Topology Construction

The topology construction algorithm seems to work quite well on random node placements, also for a high number of nodes. But for grid node placements the minimal capacity reachable (Fig. 4.1) is always worse than the capacity value of a comparable (regarding the number of nodes) random node placement. This could be because in the grid placement, every node has only up to 4 possible neighbors to chose as routing link. In a random placement some nodes may have less neighbors, which doesn't leave much choice, while other nodes have more potential neighbors, which can be used to balance the distribution of nodes to gateways.

The bad results of the topology construction in large grids are probably mainly due to the balancing heuristic. If a node wants to connect to a gateway that has already 12 connected nodes, it attempts to connect to a second gateway with less than 12 routers connected. And all the routers on the path to the second gateway, that may already be connected to a third gateway, are also connected to the second gateway, including all the routers connecting to the third gateway over these routers. (see also Fig. 6.1)

A bad topology construction can have a big impact on the performance of a network. The bottleneck is almost always at the last hop to the gateway, and too many nodes connected to one gateway make this bottleneck even tighter. This limits the throughput in a way, that the channel allocation algorithm can't improve the throughput much, sometimes even compared to a random channel assignment. For these reasons it is important to have a good topology construction algorithm.

6.2.1 Improvements

As a first possibility to evaluate, we could remove the assumed improving balancing part of the topology construction to see if it is not even worsening the result.

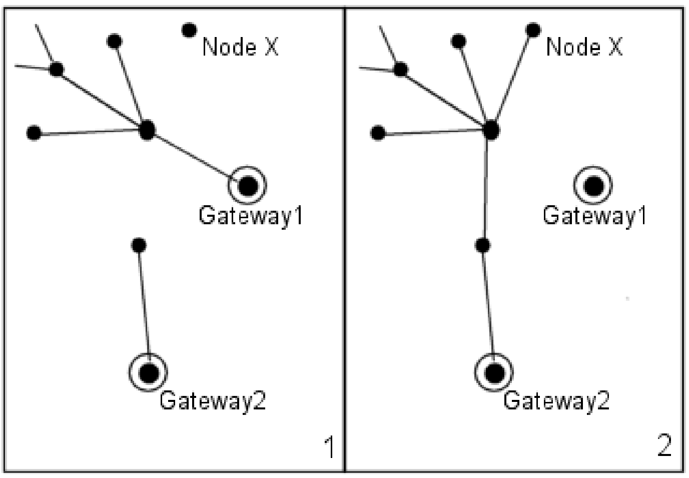


Figure 6.1: Node X is the next node to connect to a gateway. Naturally it would chose Gateway1 to connect to. Assuming Gateway1 has already 12 nodes connected, Node X decides to connect to Gateway2, to not overloading Gateway1. But as the path of Node X to Gateway2 leads over a node already connected to Gateway1, this node and all nodes connected to it, get reconnected to Gateway2, which in the end has 14 nodes connected.

We can also do a minor improvement of the upper bound of the minimal capacity of a scenario class. We should divide the link capacity by the rounded up ratio of the number of nodes divided by the number of gateways if the ratio is no integer value. Instead of always dividing by the ratio.

6.3 General

As an alternative to fixing the ratio between interference range and communication range to three, we could also use a continuous model. Define the interference value up to the communication range as 100% and fully count the traffic flows to the collision domain. If there is another link which interferes just with 50% of the communication range value, only 50% of the traffic flows of this link are added to the collision domain traffic flows.

References

[1] "The official website for wimax," *http://www.wimaxforum.org*.

[2] A. Raniwala, K. Gopalan, and T. Chiueh, "Centralized channel assignment and routing algorithms for multi-channel wireless mesh networks," *ACM Mobile Computing and Comm Review (MC2R)*, no. April'04, 2004.

[3] A. Raniwala and T.-c. Chiueh, "Architecture and algorithms for an ieee 802.11-based multi-channel wireless mesh network," in *Infocom*, 2005.

[4] M. Alicherry, R. Bhatia, and L. (Erran) Li, "Joint channel assignment and routing for throughput optimization in multi-radio wireless mesh networks," in *MobiCom*, (Cologne, Germany), 2005.

[5] T. R. Jensen and B. Toft, *Graph Coloring Problems.* Wiley Interscience, New York, 1995.

[6] H. T. Friis, "A note on a simple transmission formula," *Proc. IRE*, vol. 34, no. 5, pp. 254–256, 1946.

[7] "ns-2 documentation: Radio propagation models," *http://www.isi.edu/nsnam/ns/doc/ns_doc.pdf*, pp. 186–190.

[8] J. Tang, G. Xue, and W. Zhang, "Interference-aware topology control and qos routing in multi-channel wireless mesh networks," in *MobiHoc*, 2005.

[9] R. K. Jain, D.-M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *DEC Research Report, Tech. Rep.*, vol. Sept, 1984.

[10] "ns-2 network simulator," *http://www.isi.edu/nsnam/ns/*.

[11] "Opnet network simulator," *http://www.opnet.com/*.

[12] "Qualnet network simulator," *http://www.scalable-networks.com/*.

[13] M. K. Marina and S. R. Das, "A topology control approach for utilizing multiple channels in multi-radio wireless mesh networks," in *Broadnets*, 2005.

[14] A. K. Das, H. M. K. Alazemi, R. Vijayakumar, and S. Roy, "Optimization models for fixed channel assignment in wireless mesh networks with multiple radios," in *SECON*, 2005.

[15] M. Kodialam and T. Nandagopal, "Characterizing the capacity region in multi-radio multi-channel wireless mesh networks," in *MobiCom*, (Cologne, Germany), 2005.

[16] A. Raniwala and T. Chiueh, "Evaluation of a wireless enterprise backbone network architecture," in *Hot-Interconnects*, 2004.

[17] J. So and N. H. Vaidya, “Routing and channel assignment in mult-channel multi-hop wireless networks with single-nic devices,” tech. rep., University of Illinois at Urbana-Champaign, 2004.

[18] H.-Y. Wei, S. Ganguly, R. Izmailov, and Z. J. Haas, “Interference-aware ieee 802.16 wimax mesh networks,” in *Vehicular Technology Conference*, 2005.

[19] P. Kyasanur and N. H. Vaidya, “Routing and interface assignment in multi-channel multi-interface wireless networks,” in *WCNC*, 2005.

[20] B. Raman, “Channel allocation in 802.11-based mesh networks,” in *Infocom*, (Barcelona, Spain), 2006.

[21] B. Raman and K. Chebrolu, “Revisiting mac design for an 802.11-based mesh network,” in *HotNets-III*, 2004.

[22] B. Raman and K. Chebrolu, “Design and evaluation of a new mac protocol for long-distance 802.11 mesh networks,” in *Mobicom*, Aug/Sep 2005.

[23] T. Vanhatupa, M. Haennikaeinen, and T. D. Haemaelaeinen, “Optimization of mesh wlan channnel assignment with a configurable genetic algorithm,” in *WiMeshNets*, (Waterloo, Ontario, Canada), 2006.

[24] W. Klotz, “Graph coloring algorithms,” *Mathematikbericht, TU Clausthal*, vol. 5, pp. 1–9, 2002.

[25] J. Culberson, “Joseph culberson’s graph coloring resources page, http://web.cs.ualberta.ca/ joe/coloring/,” 2006.

[26] R. Brown, “Chromatic scheduling and the chromatic number problem,” *Manage. Sci.*, vol. 19, no. 4, pp. 451–463, 1972.

[27] M. Kubale and B. Jackowski, “A generalized implicit enumeration algorithm for graph coloring,” *Communications of the ACM*, vol. 28, no. 4, pp. 412–418, 1985.

[28] I. Juhos, A. Toth, M. Tezuka, P. Tann, and J. I. v. Hemert, “A new permutation model for solving the graph k-coloring problem,”

[29] I. Juhos, A. Toth, and J. I. v. Hemert, “Binary merge model representation of the graph colouring problem,” 2004.

[30] “Php documentation,” *http://www.php.net*, 2006.

[31] “Gd graphics library website,” *http://www.boutell.com/gd/*, 2004.

Appendix A

Evaluation with Three Channels

For the evaluation of the algorithms with three channels to allocate we have a reduced number of algorithms. The merge algorithms fails to allocate the channels in 39 of the 40 evaluated scenario instances, therefore it is not included in the following evaluations. The Tabu Greedy algorithm is only applicable for scenarios with a smaller number of gateways than channels. As we have three channels, we consider only the scenarios Rand12, Rand24 and Grid25 for the evaluation of the Tabu Greedy Algorithm. Greedy Breadth First, Greedy Most Interfered First and Random are evaluated for all the scenarios.

A.1 Capacity

In the capacity overview (Fig. A.1) we can see that Tabu Greedy has a high average capacity and a close to the upper bound minimal capacity. But these values are not very meaningful, because for Tabu Greedy the average values are just from three scenarios, among them Rand12 and Rand24 where all algorithms have good values.

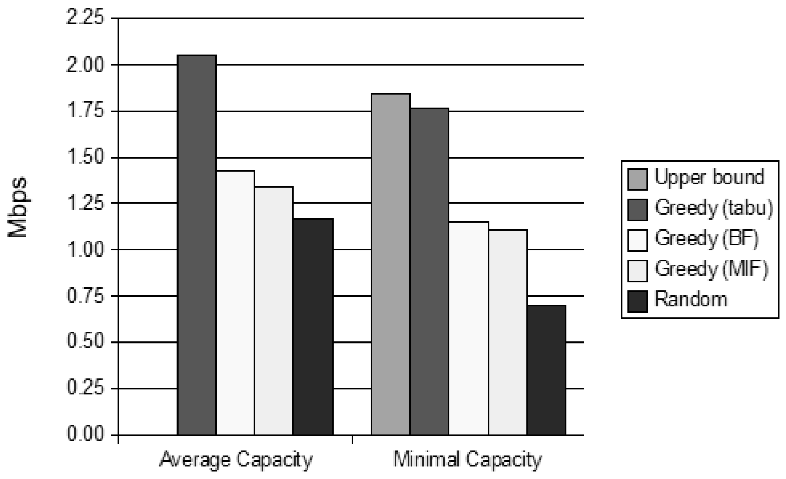


Figure A.1: Average values for the average and the minimal capacities for three channels, over all scenario instances grouped by algorithm. Along with the upper bound for minimal capacity determined by the topology construction.

If we just consider the scenarios where we evaluated all the algorithms, there are still differences in the average capacity, but they are smaller. Greedy Breadth First reaches 94% and Greedy Most Interfered First reaches 88% of the average capacity of Tabu Greedy. Concerning the minimal capacity all greedy algorithms are equal.

Looking at the detailed results for every scenario in Fig. A.2 for the average minimal capacity and in Fig. A.3 for the average capacity, we see that with ascending number of nodes, the capacities drop rapidly, and stagnate for node numbers higher than 36.

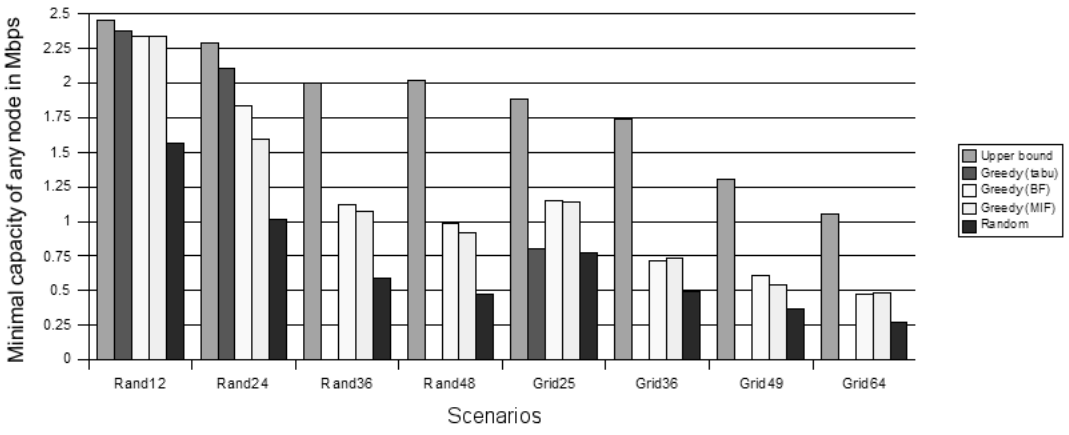


Figure A.2: Average Minimum Capacities for three channels, over all instances of scenarios

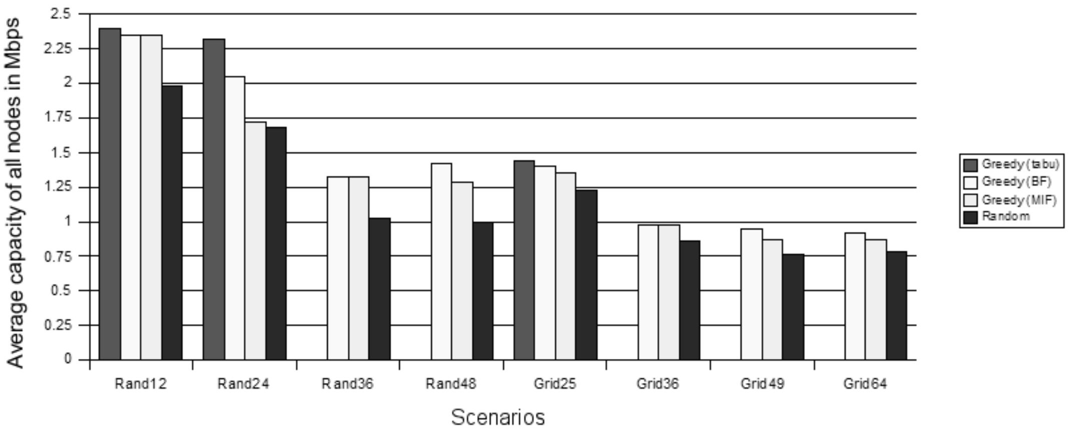


Figure A.3: Average Capacities for three channels, over all instances of scenarios

A.2 Fairness

The fairness results with three channels are, as opposed to the capacity or interference results, expected to be better than the results with 12 channels, and they are. We will now have a short look at the average fairness values over all scenarios of all algorithms. (Tab. A.1)

We see that for each fairness criteria, the values are about 0.9. It's not surprising, that the values are better than with 12 channels. Because the groups of nodes, links or flows on one channel become larger and therefore, little variation has a smaller consequence on the fairness value.

	Fairness Value
Nodes per Channel	0.91
Flows per Channel	0.87
Links per Channel	0.89

Table A.1: Average fairness indices of the distribution of nodes, traffic flows and links on channels over all scenarios and all algorithms.

A.3 Interference

In Tab. A.2 we see the average interference and the average maximal interferences for each algorithm. For Tabu Greedy, again only Rand12, Rand24 and Grid25 are considered.

Allocation Algorithm	Average Interference	Average Maximum Interference
Greedy (Tabu)	653.98E-08	1138.88E-08
Greedy (BF)	1.55E-08	8.61E-08
Greedy (MIF)	1.52E-08	7.99E-08
Random	519.34E-08	1718.10E-08

Table A.2: Average and average maximum interference values over all scenarios, grouped by algorithm

We see that Tabu Greedy now produces similar interferences as a random allocation. But if we consider the fact, that for two of the three evaluated scenarios (Rand24 and Grid25), Tabu Greedy assigns the same channel to 22 or 23 nodes, its astonishing, that the interference values aren't higher.

The interference values in general are about 10 times higher than with 12 channels.

Appendix B

Implementation

This chapter shows what have been implemented and how it was implemented.

B.1 Files

In the first section we describe the file structures of the files generated and used for the evaluation.

B.1.1 Scenario file

The first block of numbers contains the same values as the second block, the first block is for easier parsing and the second block is for easier human reading, with explanations, what every value signifies.

The third block is information about the topology construction, it indicates for every gateway (GW) the number of nodes connected to it, including itself.

```
200
200
12
1
115
10
12

Square x: 200m
Square y: 200m
Nodes: 12
Gateways: 1
Communication range: 115m
MinDist: 10m
Max Nodes/Gateway: 12

GW1,12
```

The filename of a scenario file is constructed as follows:

```
<scenario name>.sce
```

B.1.2 Nodes file

Each line in the .nodes file gives information about a node. The first value is the unique identifier, the second value the x-coordinate, the third value the y-coordinate and the fourth value is the z-coordinate.

```
1 152 95 5
2 19 88 5
3 152 131 5
...
```

The filename of a nodes file is constructed as follows:

`<scenario name>.nodes`

B.1.3 Edges file

Each line of the .edges file stands for an edge in the communication graph. The two values in a line are node identifiers, where the first value stands for the node subscribing with his subscriber station interface and the second value stands for the parent node. This means the subscriber station interface of the first node is tuned to the same channel as the base station interface of the second node.

```
2 10
3 1
4 1
5 1
...
```

The filename of a edges file is constructed as follows:

`<scenario name>.edges`

B.1.4 Color file

In an color file each line stands for an allocation of a color/channel to a nodes base station interface. The channel of the subscriber station interface is determined by the base station interface of the node it is subscribed to. The first value in the color file is a node identifier and the second value is the index of the allocated color.

```
1 0
2 1
3 1
4 2
...
```

The filename of a color file is constructed as follows:

`<scenario name>.color.<number of allocated colors>.<allocation algorithm>`

B.1.5 Evaluation file

For every color assignment file, we can create an evaluation file. The first block in this file indicates the scenario name, the applied algorithm and the number of colors assigned.

In the second block some Fairness values are already computed. The Fairness of the distribution of traffic flows on the channels/colors, of the distribution of nodes to colors and of the distribution of links to colors.

The third block indicates for every node the number of traffic flows in the collision domain of its bottleneck link.

The fourth block indicates for every node the additive interferences it gets from other interfaces on the same channel.

```
 scenario,12_4,
 algorithm,greedy,
 colors,3,

 FlowsPerColorFairness,0.518519,
 NodeColorFairness,0.774194,
 LinkColorFairness,0.584541,

 bottleneckLinkFlow,1,0
 bottleneckLinkFlow,2,22
 bottleneckLinkFlow,3,22
...

 nodeInterference,1,0
 nodeInterference,2,1.12982e-08
 nodeInterference,3,9.86915e-09
 ...
```

Similar to the color filename, the filename of a evaluation file is constructed as follows:

```
<scenario name>.eval.<number of allocated colors>.<allocation algorithm>
```

B.2 Scenario Generator

There are two different scenario generator, one for random placements and the other for grid patterns. They differ in the parameters they take and in the node placement, but for the topology construction the same algorithm is used.

The random scenario generator is called by

```
generate <Scenario name> <x_max> <y_max> <numNodes> <numGateways>
  <minDist> <maxDist> <max Nodes/Gateway>
```

Where `<Scenario name>` is a freely eligible identifier for the scenario, `<x_max>` and `<y_max>` are the maximum coordinate values, `<numNodes>` the number of nodes (routers AND gateways), `<numGateways>` the number of gateways, `<minDist>` the minimal distance in meters that have to be respected between two nodes and `maxDist` the maximal distance in meters the nearest neighbor may have. `<maxDist>` can be set to the communication range, to perform a first check, if the nodes are connectable with a given communication range. `<max Nodes/Gateway>` indicates the maximal number of nodes that may connect to one gateway.

The grid scenario generator is called by

```
generategrid <Scenario name> <xNodes> <yNodes> <numGateways> <dist>
   <communication range>
```

Where `<xNodes>` and `<yNodes>` signify the number of nodes in x- and y-direction that span the grid. `<dist>` is the horizontal and vertical distance between two nodes in the grid in meters and `<communication range>` is the communication range in meters.

B.2.1 Node Placement

Random

The input parameters of the random node placement module are: The number of nodes to be placed, the minimal and maximal x- and y-coordinate, the minimal distance that have to be respected between two nodes and the maximal distance the nearest neighbor may have. It also gets the filename where to save the .nodes file.

The placement module iteratively computes a random placement of the indicated number of nodes and checks it for the distance constraints until it gets a placement that meets the constraints. Then it writes the nodes file.

Grid

The input parameters for the grid placement module are: The number of nodes in x- and y-direction, the number of gateways, the distance between the nodes and the nodes filename where to write the placement.

The module then choses a random position on the grid for every gateway. If a randomly generated position is already occupied by another gateway, it computes new coordinates.

The other nodes are placed sequentially on the grid, starting at (0,0) and skipping every position of a gateway.

B.2.2 Topology Construction

We implicitly assume that the first `<numGateways>` nodes in the nodes file are the gateways. For the random placement this doesn't matter, because all the nodes including gateways were placed randomly. And in the grid placement module, we chose the gateway positions first and write them into the nodes file at the first `<numGateways>` positions.

The inputs for the topology construction are the number of gateways, the nodes information including the number of nodes, the communication range and the maximal number of nodes per gateway. And the edges filename, where to save the edges file.

For every gateway, a virtual broadcast is performed, meaning its potential links are followed and for every node the minimal distance and outlink for every reachable gateway are stored.

If there is any node with no reachable gateway, the topology construction fails and the node placement is called again.

Then the nodes are connected to the gateways according to the algorithm described earlier.

If the topology construction fails to construct a topology within the constraints, it falls back into the generator module, where the node placement is called again.

B.3 Channel Allocation

The channel allocation algorithms are compiled together in the assign-application, which is called as follows

```
assign <scenario name> <numColors> <allocation algorithm>
```

Where `<numColors>` is the number of colors to be assigned and `<allocation algorithm>` $\in$ `{greedy, a-greedy, tabu, merge, random}`. Where `greedy` means greedy breadth first and `a-greedy` ('adaptive greedy') means greedy most interfered first.

B.4 Evaluation

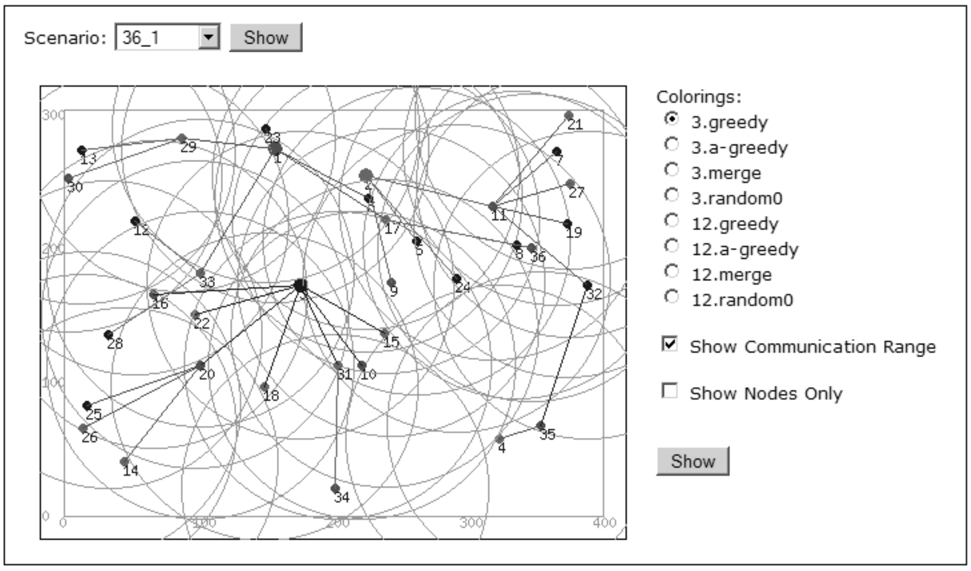
To evaluate an assignment and write the according evaluation file, call

```
evaluate <scenario name> <numColors> <allocation algorithm>
```

The parameters are the same as for the allocation, to identify the color file of the allocation to be evaluated. An evaluation file containing the results of the evaluation as described before is written.

B.5 Visualizer

As a by-product of this thesis, a visualizer for scenarios has been implemented. It is written in PHP [30] using the graphics library [31] and reads scenario files, nodes files, edges files and different color files to display the scenario with different (or without) colorings in a PNG image file.


Scenario: 36_1

Colorings: [✓ 3.greedy] 3.a-greedy 3.merge 3.random0 12.greedy 12.a-greedy 12.merge 12.random0

[✓ Show Communication Range]

Show Nodes Only

Figure B.1: Screenshot of the Visualizer.

To display a scenario, one just has to copy all the files into a folder on the web-server, named after the scenario. The visualizer searches each folder for a scenario file, and if there is one, the visualizer displays the according scenario.

Appendix C

Graph Coloring and the joint topology construction and channel allocation problem

A graph coloring algorithm assigns colors or consecutive integers respectively to certain objects in a graph like vertices, edges or faces. Vertex coloring is the most common case, and every other case can be transformed into a vertex coloring case. But in some cases, edge coloring or face coloring might be a better model to understand a certain problem. The problem of finding a minimum coloring of a graph is NP-hard. The problem of deciding, if there is a coloring that uses at most k colors, is NP-complete.

In the following section, we tried to solve the joint topology construction and channel allocation problem with graph coloring.

C.1 Vertex Coloring

Given a graph $G = (V, E)$, and a function $f : V \rightarrow N$, f is a vertex coloring of G. f is called a proper coloring, if no two adjacent vertices are assigned the same color. Or for every two adjacent vertices v_1 and v_2, holds $f(v_1) \neq f(v_2)$. (Fig. C.1)

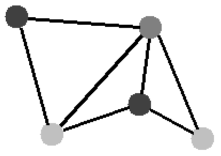


Figure C.1: Graph with Vertex Coloring in 3 Colors

G is k-colorable, if there is a proper coloring of G with k colors. The least number of colors needed to color a graph is called its chromatic number $\chi(G)$.

C.2 Edge Coloring

Given a graph $G = (V, E)$, and a function $g : E \rightarrow N$, g is a edge coloring of G. g is called a proper coloring, if no two adjacent edges are assigned the same color. Or for every two adjacent edges e_1 and e_2, holds $g(e_1)! \neq g(e_2)$. (Fig. C.2)

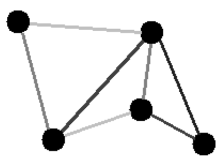


Figure C.2: Graph with Edge Coloring in 4 Colors

C.3 Mapping and Adapting common Graph Coloring to a Channel Allocation Problem

In its original form, a graph coloring algorithm is performed on a given graph, and it assigns colors to the nodes, trying to minimize the number of used colors. Our problem is different, we have a fixed number of colors/channels and we can use them all, but only these colors. As we need to assign channels to connections, it might be clever to map the assignment to an edge coloring problem. On the other side, we have to assign one channel to each node or his BSI respectively, thus it might also be a good idea to map it to an vertex coloring problem.

C.3.1 Vertex Coloring

Mapping our problem to a vertex coloring problem then seems easy. But we have to modify it in a way, we can color a node with multiple colors for multiple interfaces. Every interface (BSI or SSI) gets a color (channel). The color of the SSI is inherited from the BSI it connects to. And for the assignment of a color to the BSI, the algorithm can chose any color not yet used on any (BSI or SSI) interface of one of its neighbors.

To model the allocation problem as a vertex coloring problem, we model all the interfaces (base station and subscriber station) as vertices of the graph. An edges in the graph model binary interference links, meaning interference is either 0% or 100%.

Additional Constraints:

- subscribed BSI/SSI must have the same color (chose color of SSI according to the color of the BSI)

Constrained vertex coloring seems simple, but it is also quite anti intuitive because of the only implicit connections and the splitting of nodes into interface nodes. Also it fails to model some important facts, as also the edge coloring model in the following paragraph.

- The fact that links nearer to the gateways need higher bandwidths, because traffic is accumulated can't be easily modeled with graph coloring.
- In general, graph coloring fails to model continuous interferences. Interference in graph coloring is always either 0% or 100%.

C.3.2 Edge Coloring

As in vertex coloring, we model all the interfaces as vertices of the graph and an edge in the graph models a binary interference link.

Additional Constraints:

- same color for all edges at BSI vertices

- only one colored edge at SSI vertices
- only colored edges between a BSI and a SSI vertex are allowed
- "incoming" edges to a BSI vertex mustn't have the same color as "incoming" edges to a neighboring BSI vertex

Constrained edge coloring could be a good, intuitive model for our specific problem. But also edge coloring with constraints fails to model some facts.

C.3.3 Total Coloring

In total coloring, the vertices of the graph G model the physical nodes of our WMN. Vertices A and B are connected by an edge, if node A can hear node B and vice versa.

We now have to color both, vertices and edges. The color of a vertex stands for the channel assigned to the BSI of the corresponding node in the WMN. If an edge is colored, its color stands for an established communication channel between the adjacent nodes. Basically we model channels assigned to BSIs as vertex coloring and channels assigned to SSIs as edge coloring.

For the coloring, an algorithm has to consider the following constraints:

For coloring the vertices:

- chose from colors that are not used by neighboring vertices or by colored edges adjacent to neighbors

For coloring the edges:

- the number of colored edges in one color adjacent to a node colored in the same color is unlimited
- the number of colored edges in one color adjacent to a node with a different color is limited by the number of SSI a physical node has i.e. 1 or 2.

Invariant:

- With GC defined as subgraph of G, having all the vertices and colored edges of G, GC has to be connected.

C.3.4 Reversed Heuristics

Sequential algorithms for graph coloring always sort the vertices, i.e. They chose the order in which the vertices gets colored. This is not applicable in our problem, because we have to color the joining node. We can only use a simple algorithm like always chose the smallest color available for the actual node. But we can also make use of Heuristics for the Graph coloring problem.

There are heuristics, determining which node to chose next for coloring with the smallest color. The node with the most neighbors (LF - Largest First) or the node with the most different colors at the vertices adjacent to it (DSATUR - Degree of Saturation). That means, the node with the least neighbors or the least different colors adjacent to it is chosen at last. So we can use reversed versions of these heuristics, saying: If we have to change the color of a neighbor, because in the actual constellation there is no possibility to chose a color, chose the neighbor with the least neighbors (revLF) or the neighbor with the least different colors at the vertices adjacent to it (revDSATUR).

Appendix D

Pseudocode of Algorithms

D.1 Topology Construction

```
// Compute matrix with all possible links,
//    i.e. with distance < communication range

int** link = new int[numNodes][numNodes];
for(int i=0; i<numNodes; i++)
{
 for(int j=0; j<numNodes; j++)
 {
    if(distance[i][j]<maxDist)
    {
      link[i][j] = 1;
    }
    else
    {
      link[i][j] = 0;
    }
   }
}


// compute gatewayData

for(int i=0; i<numGateways; i++)
{
   "broadcast" from Gateway i and store minimal distance at each node;
}

// if there is any node with no reachable gateway, exit

if(<any node>.numGwReachable == 0)
    {return false;}


// connect nodes that have only one reachable Gateway
```

```
for(<any node>.numGwReachable == 1)
    {connect node to gateway;}

// connect other nodes, with decreasing max(minDistToGw)
//    considering a maximum of 12 nodes to a GW

while(still nodes to connect)
{
    // determine next node
    maxMinDist = 0;
    nextNode = -1;

    for(int i=0; i<numNodes; i++)
    {
        if(nodes[i].gatewayId == -1)
        {
           if(nodes[i].minDistToGw > maxMinDist)
           {
              nextNode = i;
              maxMinDist = nodes[i].minDistToGw;
           }
        }
    }

    // check if gateway is overloaded (soft upper bound)

    if(gwConnectionCount[gwIndexToConnect] > 12)
    {
        // find minimum distance gateway among other reachable gateways
    }

    connect node to gateway;
}
```

D.2 Greedy (Breadth First)

```
/* Input: connection matrix edge[node1][node2] edge from node1 to node2
          interference matrix interference[node1][node2] interference on
             link from node1 to node2 */


color[i][c] /* 1: BS of node i has color c */
array nodeorder[n] /* order for greedy search */

/* Order vertices breadth first */
nodeorder[0] = root
nextFill=1
act = 0
while(nextFill<n)
{
  for(i=0, i<n, i++)
  {

    /* if node i is subscribed to nodeorder[act] */
    if(connected[i][nodeorder[act]])
    {
      nodeorder[nextFill] = i
      nextFill++
    }
  }
  act++
}

/* go through nodes and assign colors greedy */
while(i<n)
{
  array colorInterferences[c] initialised with 0

  /* sum up all interferences for each channel */
  for(j=0, j<n, j++) /* iterate nodes */
  {
    for(k=0, k<c, k++) /* iterate colors */
    {
      if(color[j][k]==1) /* get the color of the node */
      {
        colorInterferences[k] += interference[nodeorder[i]][j]
         /* add the interference with this node to the color array */
      }
     }
  }

  chosenColor = 0

  /* choose color with minimal interference */
  for(k=0, k<c, k++)
  {
    if(colorInterferences[k] < colorInterferences[chosenColor])
    {
      chosenColor = k
    }
  }

  /* assign color with minimal interference */
  color[nodeorder[i]][k] = 1;

  i++;
}
/* reached the end, i=n, found a solution */
```

D.3 Merge

```
main()
{
  start with nxn matrix MT as defined above
  m := n /* m is actual number of rows*/
  while(m>12) /* assuming there are only 12 channels */
  {
      /* row numbers of the possible merge with minimal interference */
      min_i, min_j := 0

      /* interference of the possible merge with minimal interference */
      min_inter := MAX

      for(i=0, i<m, i++)
      {
           for(j=0, j<i, j++)
           {
            /* if found a better solution */
                if(checkToMerge(i, j) < min_inter)
                {
                     min_i := i; min_j := j; min_inter := checkToMerge(i, j)
                }
           }
      }
      merge(min_i, min_j);
      m--;
   }
}

float checkToMerge(i, j)
{
    interference := 0

    /* for row i and j check every column s */
    for(s=0, s<n, s++)
    {
          if(MT[i][s]==T && MT[j][s]==T) /* -> cancel merge */
          {
                interference := MAX
                exit
          }

          /* only consider channels one of the nodes is tuned to */
          if(MT[i][s]==T)
          {
                interference = interference + MT[j][s]
          }
          if(MT[j][s]==T)
          {
                interference = interference + MT[i][s]
          }
    }
}
```

```
merge(i, j)
{
    /* i and j are merged into row j, as j is always smaller that i*/

    /* for row i and j merge every column s */
    for(s=0, s<n, s++)
    {
          if(MT[i][s]==T || MT[j][s]==T)
          {
                MT[j][s] = T
          }
          else
          {
                MT[j][s] = MT[j][k] + MT[i][k]
          }
    }
}
```

Appendix E

Assignment

E.1 Introduction

Wireless mesh networks are networks that create connectivity over multiple wireless hops. With IEEE 802.16 (WiMax) technology maturing within the next 2-3 years, wireless mesh networks become an interesting alternative to wired access in metropolitan areas (see figure E.1 for an illustration). Moreover, communication systems that base on wireless mesh networks are of particular interest to public safety applications, e.g. to create network connectivity right after a disaster such as an earthquake or a tsunami. To develop these systems one needs to find algorithms (i) to construct a mesh topology from a set of given nodes and (ii) to allocate N available reusable orthogonal frequencies to the links in the topology. This topology construction and frequency allocation should be done in a way that meets traffic demand and minimizes quality of service degradation due to interference of multiple wireless links that use the same frequency.

In the TOWN project, we search for algorithms that can be employed in public safety applications where telephony is the primary application. Thus, traffic demand is fixed and predictable. Since interface cards working in full mesh mode will likely not be available within the next 2-3 years, the TOWN project heads toward developing algorithms and protocols for frequency allocation assuming that network nodes are equipped with two or more interfaces. The first interface works in subscriber station mode to connect the node to the network. The second interface works in subscriber station mode to offer network connectivity to other nodes. This interface configuration leads to point-to-multipoint communication relations. Moreover, the focus of the TOWN project is on scenarios where network nodes are stationary, i.e. do not move. Since the original assignment for this thesis turned out to be too broad, this revised assignment recommends (i) to focus on the frequency allocation problem for a given mesh network spanning tree topology for the two interface case and (ii) to assume that no nodes join or leave the network.

E.2 Assignment

E.2.1 Objectives

The revised overall objective of this master thesis assignment is to focus on developing an algorithm for frequency allocation for a given mesh network spanning tree topology as depicted in figure E.1. This algorithm may be based on some heuristic. The input to the algorithm is the complete set of potential communication qualities and interferences as obtained by measurements. The output is the channel alloca-

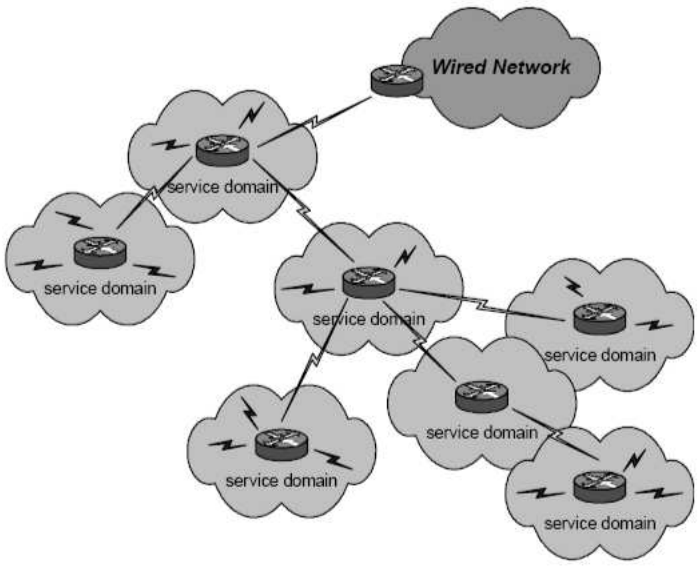


Figure E.1

tion. Interference between multiple base station interfaces operating on the same frequency can be assumed to be additive. The developed algorithm needs to be evaluated and - if possible - compared to the optimal channel allocation, which can be determined by searching through all possible allocations.

E.2.2 Tasks

- Understand the problem. Clarify and refine the problem statement. Identify how the input to the algorithm can be coded. Eventually, break the before mentioned problem into a set of smaller problems. Check the complexity of the problem (NP-hard, etc.).
- Study related work. Starting points are the papers on channel allocation, mesh networks and multi-radio networks published in INFOCOM, SIGCOMM and MOBICOM conferences, particularly the papers listed below in the reference section.
- Propose an algorithm for frequency allocation.
- Identify evaluation criteria and parameters to assess the performance of the algorithm.
- Set up a QualNet simulation, implement and evaluate your algorithm.
- Write up your simulation results into a thesis and document your work.

E.3 Proposed Schedule

To cope with the assignment in the remaining time, we propose the following schedule.

The student is hereby asked to review this schedule. Revisions of the schedule may be suggested and discussed with the advisors until July 25th, 2006. Otherwise, we assume that the student agrees with the proposed schedule. Moreover, we note that

Remaining time (weeks)	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reviewing the Problem	X													
Related Work	X													
Proposing Algorithms		X	X	X										
Evaluation Metrics				X	X									
Setting-up Simulations					X	X								
Implementation						X	X	X						
Simulations and Evaluations										X	X	X		
Writing Thesis	X				X						X	X	X	X
Final Presentation												X	X	
Cleaning up													X	

- It is the students responsibility to make sure that the work progresses according to schedule. If it is not possible to solve tasks according to schedule, advisors have to be informed asap with clearly specifying the problem(s).
- At the end of each task specified in the schedule, the student has to write up and deliver a ten line summary of the task. This write up can be done in the thesis tikiwiki page.

E.4 Deliverables and Organization

- If possible, student and advisor meet or telephone on a weekly basis to discuss progress of work and next steps. If problems/questions arise that can not be solved independently, the student should not hesitate to contact the advisor anytime.
- In regular intervals (e.g. every two months) intermediate reports are due. These reports are linked to short presentations of 15 minutes to the professor and the advisors. In these presentations, the student has to discuss major aspects of the ongoing work including results, problems, and remaining work.
- At the end of this thesis, a presentation of 15 minutes must be given either in teleconference or in the communication systems group meeting. The presentation should carefully introduce settings and background of the work. Moreover, it should contain an overview of the major results and conclusions from the work.
- We encourage writing all reports in English. However, reports can also be written in German. The final report must contain a summary, the assignment and the time schedule. Its structure should include an introduction, a methods/design section, a results section and a conclusion section. Moreover, the final report should include a complete documentation of all produced software. Related work must be correctly referenced. See http://www.tik.ee.ethz.ch/∼flury/tips.html for more tips on thesis writing. Three hard copies of the final report must be delivered to TIK.
- Any software which is produced in relation with this thesis needs to be delivered before ending the thesis. This includes all source code and documen-

tation. The source code may then be published as open source. Moreover, the PDF and the source code employed to generate the final report also have to be delivered. This includes data to draw the figures. Preferred format for delivery is a CDROM.

E.5 References

1. A. Raniwala, K. Gopalan, T. Chiueh, ”Centralized Channel Assignment and Routing Algorithms for Multi-channel Wireless Mesh Networks”, ACM Mobile Computing & Comm Review (MC2R), April ’04.
2. A. Raniwala, T. Chiueh, ”Evaluation of A Wireless Enterprise Backbone Network Architecture”, Proc. of the 12th Hot-Interconnects ’04.
3. A. Raniwala, T. Chiueh, ”Architecture and Algorithms for an IEEE 802.11-Based Multi- Channel Wireless Mesh Network”, Proc. of IEEE Infocom’05, March 2005.
4. P. Kyasanur and N. Vaidya, ”Routing and Interface Assignment in Multi-Channel Multi- Interface Wireless Networks,” Proc. WCNC, 2005.
5. M. Marina, S. Das, ”A Topology Control Approach for Utilizing Multiple Channels in Multi- Radio Wireless Mesh Networks”, Proc. Broadnets, Oct 2005.
6. J. Tang, G. Xue, and W. Zhang. ”Interference-aware topology control and QoS routing in multi-channel wireless mesh networks”. In Proceedings of ACM MobiHoc, pages 68–77, Urbana-Champaign, Illinois, USA, May 2005.
7. M. Alicherry, R. Bhatia and L. Li, ”Joint Channel Assignment and Routing for Throughput Optimization in Multi-radio Wireless Mesh Networks”, In Proc. Of the ACM International Conference on Mobile Computing and Networking (MobiCom), Cologne, Germany, August 2005.
8. A. Das, H. Alazemi, R. Vijaykumar, S. Roy, ”Optimization Models for Fixed Channel Assignment in Wireless Mesh Networks with Multiple Radios”, September 2005.

Appendix F

Schedule

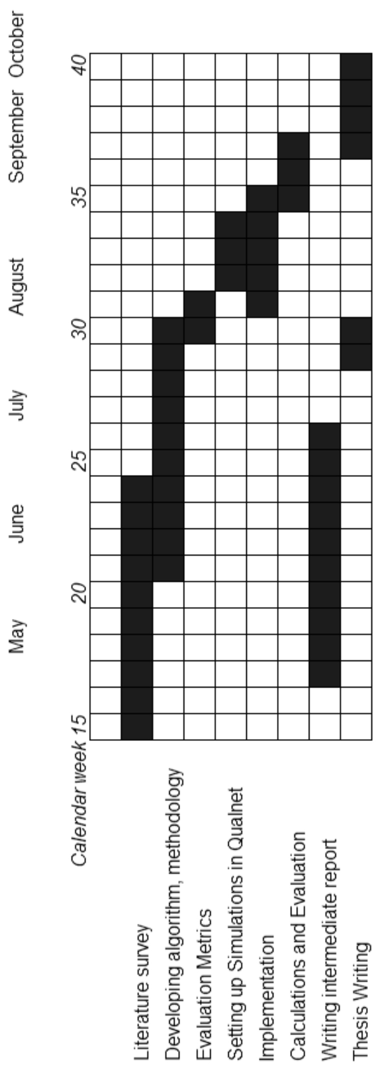


Figure F.1: Schedule