

Universität Stuttgart

Fakultät Informatik, Elektrotechnik
und Informationstechnik

Diplomarbeit Nr. 2559

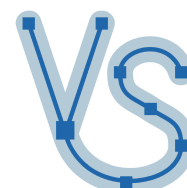
**Group formation for adaptation
purposes in wireless sensor networks**

Andreas Grau

Studiengang:	Softwaretechnik
Prüfer:	Prof. Dr. Kurt Rothermel
Betreuer:	Daniel Minder
begonnen am:	7. November 2006
beendet am:	9. Mai 2007
CR-Nummer:	C.2.1, C.2.3, C.4



Institut für Parallele
und Verteilte Systeme
Abteilung Verteilte Systeme
Universitätsstraße 38
D-70569 Stuttgart



Zusammenfassung

In den letzten Jahren wurden einige Middleware Systeme zur Adaptation von Algorithmen in drahtlosen Sensornetzen entwickelt. Ein Vertreter solcher Systeme, TinyCubus genannt, hat den Schwerpunkt auf der Anpassung von Algorithmen während der Laufzeit des Netzwerkes. Zu dieser Anpassung wird der Zustand des Netzwerkes analysiert und für jeden Knoten Parameter für optimales Verhalten ausgewählt. Auf Grund der Größe von Sensornetzen skaliert ein Ansatz, der auf individuellen Knotenparametern basiert, nicht.

Eine mögliche Lösung fasst Knoten mit ähnlichen Eigenschaften in Gruppen zusammen um die Komplexität der Suche nach optimalen Netzwerkkonfigurationen zu senken. Diese Gruppen nutzend, ist es durch eine optimale Konfiguration dieser Gruppen möglich, die Leistungsfähigkeit des Netzes zu steigern. In dieser Arbeit werden Knoteneigenschaften zur Gruppenbildung identifiziert, und darauf aufbauend werden Modelle zur Gruppenbildung entwickelt. Die entwickelten Modelle gruppieren Knoten auf der Basis von Knoteneigenschaften, wie dem Ort oder der Dichte und nutzen verteilte sowie zentralisierte Algorithmen. Während der Bewertung werden typische Netzwerktopologien, basierend auf regelmäßig und zufällig platzierten Knoten, benutzt. Simulationen dieser Szenarien zeigen, dass die Leistung von Netzwerken mit gruppenbasiert oder individuell konfigurierten Knoten nahezu identisch ist.

Abstract

In the past years, several middleware systems to adapt algorithms in wireless sensor networks have been developed. One representative of such systems, named TinyCubus, focuses on tuning the used algorithms at the runtime of a network. The adaptation analyzes the state of the network and selects best performing parameters for each node. Due to the size of sensor networks, an approach using individual parameters per node does not scale.

A possible solution groups nodes with equal properties to limit the effort to find an optimal network configuration. Based on these groups, it is possible to configure each group with an optimal setting to improve the network performance. During this thesis node properties are identified which can be used for group formation and, based on these, group formation models are developed. The developed models, grouping nodes based on properties like location or density, are using distributed and central coordinated algorithms. During the evaluation, typical sensor network topologies, with regular and randomly placed nodes, are used. The simulation of these scenarios showed that the performance of networks with group based or individually configured nodes is almost the same.

Contents

I	Introduction and Background Information	1
1	Introduction	2
1.1	Motivation	2
1.2	Purpose of Study	3
1.3	Outline	3
2	TinyCubus	5
2.1	Architecture	6
2.1.1	Design Principles	6
2.1.2	Adaptation	7
2.2	Group Formation Requirements	9
3	Related Work	12
3.1	Adaptive middleware	12
3.2	Clustering	13
3.3	Routing Algorithms	14
3.4	Monitoring Frameworks	15
4	Sensor Network Properties	16
4.1	Node Characteristics	17
4.1.1	Sensing Hardware	17
4.1.2	Data Processing Hardware	18
4.1.3	Communication Hardware	19
4.1.4	Other Hardware Devices	20
4.2	Network Topologies	20
4.2.1	Regular vs Random Structures	20
4.2.2	Static vs Mobile Topologies	22
4.2.3	Free vs Constrained Topologies	22
4.2.4	Density of Nodes	23
4.2.5	Number of Base Stations	24
4.2.6	Homogeneous vs Heterogeneous Networks	24
4.3	Application Characteristics	25
4.3.1	Network Traffic	25
4.3.2	Application Based Groups	26
4.4	Summary	26

II	Group Formation Models and Theoretical Discussion	27
5	Model Basics	28
5.1	Network Assumptions	28
5.1.1	Receiver Signal Strength Indication	28
5.1.2	Network Status Monitoring Frameworks	28
5.1.3	Mobile Nodes	29
5.1.4	Bi-directional Links	29
5.2	Fundamental Algorithms	30
5.2.1	Graph Partitioning	30
5.2.2	Cluster Head Election	32
5.2.3	Neighborhood Value	33
5.3	Optimal Group Properties	36
5.3.1	Density Consideration	36
5.3.2	Traffic Consideration	36
5.3.3	Group Borders	36
6	Group Models	38
6.1	Classification	38
6.1.1	Coordination	39
6.1.2	Used Metrics	40
6.1.3	Group Layout	40
6.2	Distributed Approach	41
6.2.1	Most Equal Neighbor	41
6.2.2	Distributed Groups - Static Table	44
6.2.3	Mobile Groups	45
6.3	Hybrid Approach	48
6.3.1	Distributed Groups - Dynamic Table	48
6.3.2	Location Based	49
6.4	Centralized Approach	53
6.4.1	Weighted Links - Graph Partitioning	53
6.5	Model Extensions	55
6.5.1	Multi Base Station Model	55
6.5.2	Environment Based Model	56
6.5.3	Different Node Types Model	57
6.6	Theoretical Model Discussion	59
6.7	Summary	62

III	Evaluation and Conclusion	63
7	Evaluation	64
7.1	Implementation	64
7.1.1	Simulator	65
7.1.2	Node Types	65
7.1.3	Node Components	66
7.1.4	Medium	69
7.1.5	Evaluation Plugins	69
7.2	Scenarios	70
7.2.1	Grid Pattern	71
7.2.2	Random Location	72
7.3	Measurement Results	73
7.3.1	No Groups - Equal Settings	75
7.3.2	Optimal Groups - Individual Settings	78
7.3.3	Group Formation Metrics	81
7.3.4	Formed Groups	83
7.3.5	Model Quality - Optimal Settings	88
7.4	Discussion	97
7.4.1	Models	97
7.4.2	Generalization	99
8	Conclusion	101
8.1	Summary	101
8.2	Limitations and Future Work	102
	List of Tables	I
	List of Figures	II
	List of Algorithms	IV
IV	Appendix	V
A	No Groups - Equal Settings	VI
B	CUBUS scenario files	VIII
C	Node positions	XI
D	Bibliography	XII
E	Statement	XV

Part I

**Introduction and Background
Information**

Chapter 1

Introduction

1.1 Motivation

During the last years a new kind of computer networks became a research object. These networks are based on dozens or even hundreds of small sensor nodes, equipped with sensor hardware to observe their environment and interconnected using wireless radio technology. The main characteristics of these so called wireless sensor networks (WSNs) [ASSC02] are the boundedness to hardware resources like CPU, memory, radio subsystem and power supply. Using their sensors these nodes can be used for several application scenarios including observing and tracking of objects [Bri07, MMLR05, MPS⁺02].

In the first years, the applications used in sensor networks were build purely on top of the operating system of the sensor nodes. The de facto standard operating system for sensor networks, TinyOS [HSW⁺00], has been developed at the University of California, Berkeley. This approach of application development requires every programmer to think about issues like routing data through the network. To conquer this problem several middleware systems have been invented [RKM02, TS05].

Another problem raises from the fact that the properties of a sensor network, like the density of nodes, are often different. The different properties require different routing protocols or different parameters of these algorithms. Adaptive middleware systems focus on this problem. The *TinyCubus* [MLM⁺05] system, developed at the University of Stuttgart, implements several algorithms for the same domain, e.g. data routing, and selects the best fitting algorithms depending on the current situation of the network.

In the current implementation of the *TinyCubus* system, the variation of parameters is based on the global situation. In other words the system analyzes the current state of the sensor network and calculates the parameters. Every node in the network reconfigures its algorithms with the same settings. This approach lacks support of sensor networks with different node properties, e.g. when the density of nodes varies in the network. The first solution, to configure every node with its own settings results is infeasible in praxis, due to the exponential parameter space. The problem with the parameter space comes from the fact that finding the optimal parameter is based on network simulation and, therefore, requires a huge effort.

Keeping the heterogeneity and the huge parameter space in mind, it becomes obvious that grouping nodes with equal properties and configuring each group with individual parameters is the best tradeoff. The formation of groups for adaptation purposes is the topic of this thesis. In the next section, the purpose of the thesis is defined in detail.

1.2 Purpose of Study

Adaptive middleware systems allow the sensor network to adapt its behavior based on the current situation of the network or the environment. The adaptation of the sensor nodes enables two approaches, a coarse and a fine granular. The coarse granular approach selects the best fitting algorithm for the current situation, e.g. selects tree or flooding based data routing. Such a selection has to be performed on each node of the sensor network and has to select the same algorithm on each node.

The focus of the fine granular approach is to tune a specific algorithm by varying parameters of the algorithm. A possible parameter for this approach is for example the number of retransmits, when routing data through the routing tree and no acknowledgment arrives. This approach allows setting different parameters on each sensor node, depending on properties of the node's environment.

In this thesis, the focus lays on the formation of groups for adaptation usage. Finding properties which can be used to form groups is subject of the first part. Based on these properties optimal group characteristics have to be defined. Using these characteristics, methods to form groups have to be modeled and evaluated. The evaluation of the methods requires the implementation of the group forming models, and the simulation using network simulators.

1.3 Outline

The remainder of this thesis is structured as follows. In general, the thesis is structured in three parts:

- Introduction and background information: Chapter 1, 2, 3, and 4
- Group formation models and theoretical discussion: Chapter 5 and 6
- Evaluation and conclusion: Chapter 7 and 8

As the topic of the thesis suggests, the groups are used for adaptation and, therefore, Chapter 2 covers the adaptive middleware *TinyCubus* [MLM⁺05], because this work serves as a theoretical foundation for a grouping component in *TinyCubus*. Since there are works related to this thesis, Chapter 3 mentions those systems. The introduction mentioned already some of the differences between classical distributed systems and the field of wireless sensor networks. Due to the importance of knowledge about the characteristics of sensor networks, Chapter 4 discusses the sensor network properties in detail.

The actual purpose of this thesis is group formation and, because of this, Chapter 5 defines optimal group properties and, based on those properties, Chapter 6 describes possible models to group nodes. The different models are classified and theoretically discussed based on the network assumptions, made by the models.

The performance of the developed models is the topic of Chapter 7. The performance evaluation is performed by developing scenarios and simulating the models using network simulators. The evaluation is finished by a discussion of the measured results. Chapter 8 concludes the thesis by summarizing the whole work.

Chapter 2

TinyCubus

Applications in sensor networks are often based on several underlying algorithms, like routing or data aggregation. For every type of algorithm several different algorithms exist and each of them makes assumptions on the network characteristics, e.g. the topology of the network and, therefore, they perform differently in a specific situation.

From the application point of view, it is irrelevant which algorithm is used, except that the algorithm performs optimally in the specific situation. To release the application developer from analyzing the network state and selecting the appropriate algorithm, a framework named *TinyCubus* [MLM⁺05] was developed. In the following sections the architecture of *TinyCubus* is discussed in detail. After that, the adaptation process is introduced, and the demand for group formation is explained.

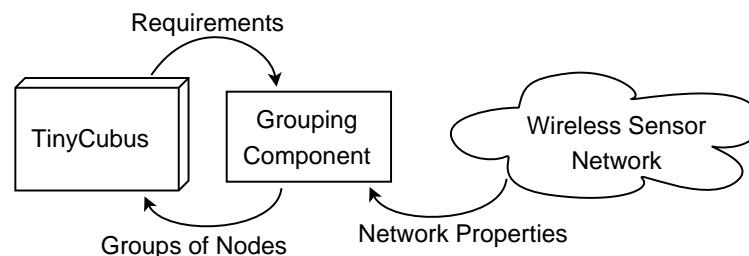


Figure 2.1: Grouping Component

Figure 2.1 shows the grouping component and its inputs and outputs. The function of the grouping component is to form groups of nodes for *TinyCubus*. To perform this task, the component needs to know on the one hand *TinyCubus*' requirements on groups and on the other hand the properties of the underlying sensor network. Section 2.2 discusses the requirements of the *TinyCubus* System.

2.1 Architecture

The following section discusses the architecture of *TinyCubus* in detail. Section 2.1.1 introduces different building blocks of *TinyCubus* and explains their functionality. Since group formation for adaptation is topic of this thesis, Section 2.1.2 focuses on the adaptation process of *TinyCubus*.

2.1.1 Design Principles

Applications developed on top of the operating system *TinyOS* [HSW⁺00] are typically written in the *NesC* programming language, which enforces a strict modular design of the applications. Figure 2.2 shows an application to observe the room conditions like humidity or temperature. The logic to detect and manage the neighborhood of a node is implemented in the *NeighborControl* component. Based on the neighborhood, the *Routing* component forms routing trees, and the *Aggregation* component calculates average values for a location.

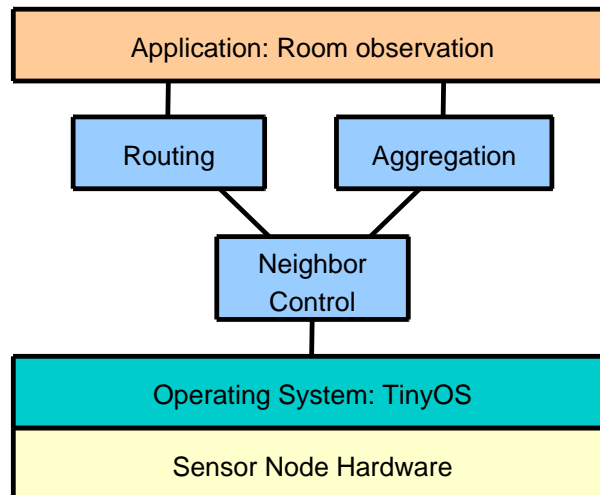


Figure 2.2: Sample application based on TinyOS

In this application the coupling between the components is explicitly modeled. When replacing one component, the information flow must be taken into account. To get a loose coupling between components, the *TinyCubus* infrastructure, shown in Figure 2.3, introduces the *Tiny Cross-Layer Framework*. This framework allows components to store their state in a repository and manages the access on it. In the room observation scenario, the *NeighborControl* component detects the neighborhood and stores the neighbors in the repository. The *Routing* component elects one of the neighbors as parent node for routing data to the base station. The *Aggregation* is now decoupled from the *NeighborControl* component, because it simply reads the neighborhood from the repository without knowledge of the origin of the data.

From another point of view, the *Tiny Cross-Layer Framework* provides the possibility of exchanging data between algorithms of different layers of the ISO/OSI network stack. Following this approach, the application (layer 7) can tell the data link layer (layer 2) to turn off the

radio module for a specific time, because the application knows that no data arrives in the next period of time. This short example demonstrates that sharing of information between different layers using the *Tiny Cross-Layer Framework* allows optimizing the performance of the sensor network, without losing modular designed applications.

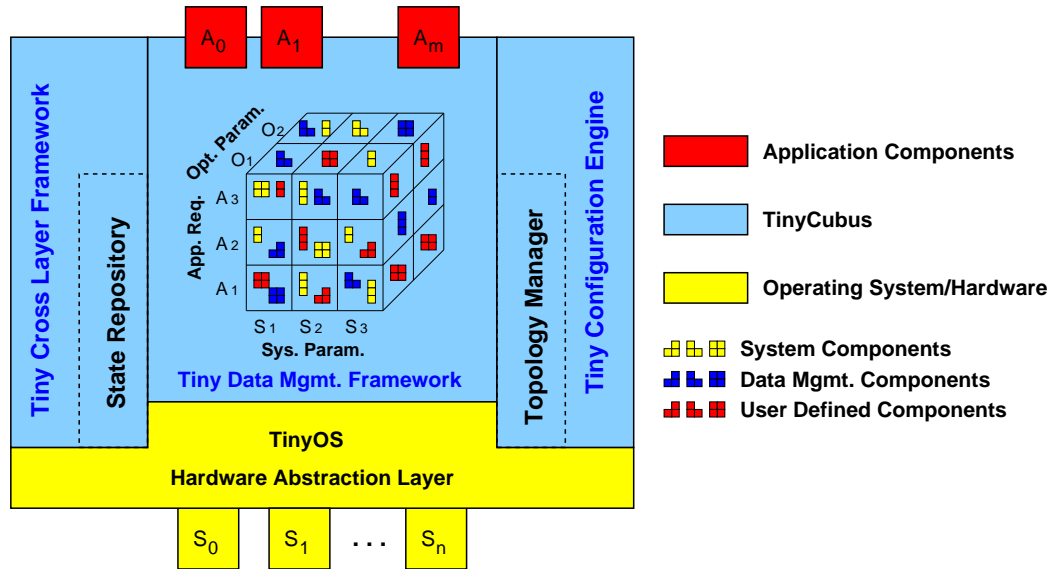


Figure 2.3: Components of TinyCubus

During the lifetime of a deployed sensor network, the conditions of the environment can change which requires to update parts of the application logic. Concerning the amount of the deployed sensor nodes, it is often infeasible to collect the nodes and install an adapted application version and redeploy the nodes. In *TinyCubus*, the *Tiny Configuration Engine* addresses this issue by allowing to install new code on running nodes. Using this component, it becomes possible to replace components like the *Routing* component with an optimized version.

Based on the loose coupling and on the possibility of replacing modules at runtime, the third element of *TinyCubus* can select an optimal component for a specific situation. The *Tiny Data Management Framework* allows the adaptation of a running application to the current network situation by installing and parameterizing an optimal algorithm. In the following section, the adaptation behavior of *TinyCubus* is discussed in detail.

2.1.2 Adaptation

Sensor networks are huge distributed systems running distributed applications. Unlike classical distributed systems the nodes in a sensor network are embedded in the environment and there is no administrator who monitors the network and reconfigures it, if there are changes in the environment or in the system. These changes are, for example, the sudden occurrence of many events in parts of the network, which require the reconfiguration or the exchange of routing algorithms to handle these events with minimal energy consumption. Another possible situation is the frequent failure of nodes in sensor network.

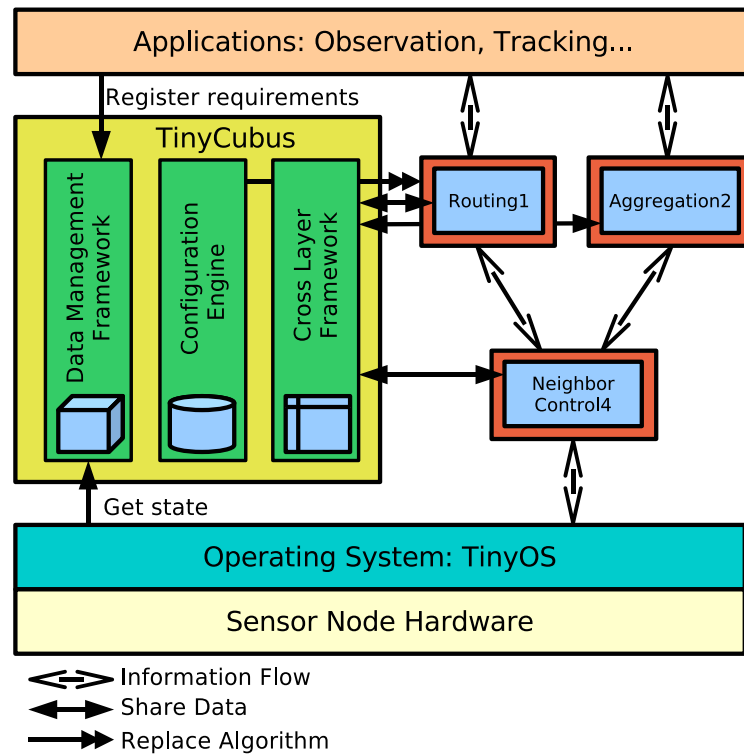


Figure 2.4: Architecture of TinyCubus

In traditional distributed systems the operator adapts the system by hand. Nodes with failures will be replaced and in situations where additionally compute power is required it is possible to deploy additional nodes. In sensor networks where no operator is available or it is impossible to replace failed nodes the network must adapt itself. In *TinyCubus* the *Tiny Data Management Framework* addresses the adaptation of the application.

Adaptation of a system to improve its behavior requires several inputs. The first is the knowledge of the network state. The adaptation component needs information about the network traffic situation or the remaining energy of the nodes. The second required inputs are the requirements of the application. This category covers the required quality of service for the application, like the needed lifetime of the network. The last input is the existence of different algorithms for a specific network functionality. These algorithms are not required to be completely different, because it is also possible to use adjustable algorithms or a combination of both approaches. Additionally to the existence of these algorithms, the influences of the algorithm and their parameters have to be known, too. To exemplify the importance of the influences, think of the following example: If an application requirement is to limit the number of transmitted data, and there exist two routing algorithms, flooding and tree based, then the system has to know which of the two algorithms requires more data to be sent.

Based on these three dimensions, the *Tiny Data Management Framework* of *TinyCubus* can select and customize the best performing algorithm. Whenever *TinyCubus* adapts the system, two of the three dimensions of the cubus are fixed. One dimension is set by the application and the other dimension is based on the current network situation. The *Tiny Data Management Framework* can choose a value, which represents the optimal configured algorithm, from the remaining dimension. In our example the tree based algorithm would be used.

Following this approach some problems occur. One raises when acquiring the state of the network. Due to the size of the network a solution where every node has to know the entire state of the network does not scale and wastes a lot of energy. Another problem is associated with the size of the cubus. If the dimensions of the cubus cover all nodes, the cardinality of the different dimensions will grow exponentially with the number of nodes and possible algorithm versions. This huge space of possible configurations makes the calculation of the network influences of different algorithms infeasible. The problem is amplified by the fact that the calculations are based on simulations of sensor networks, which require a lot of computing power.

Forming groups limits the knowledge of the network that a single node needs as well as the possible network configurations. All nodes belonging to the same group are configured with the same parameters. Following this approach, the adaptation component specifies several conditions to the group formation, which are the subject of the next section.

2.2 Group Formation Requirements

Partitioning of a sensor network into clusters of nodes is a common design pattern to get a scalable and energy-saving solution. Following this approach, the homogeneous network architecture is broken down to a hierarchical architecture where some nodes, called cluster heads, have specific roles. Typically nodes of one cluster only communicate with nodes in the same cluster or use the cluster head as a router between other clusters or the base station. A selection of different projects using clusters for data routing is introduced in the related work in Chapter 3.

When using clusters to optimize data routing the basic requirement on clusters is to reduce the energy consumption. The focus of this thesis is group formation for adaptation purposes and, therefore, the following sections identify those requirements which are necessary to use groups for adaptation. In addition to these requirements and common criteria for clustering networks, Section 5.3 defines optimal group properties for adaptation usage.

Group Nodes with Same Properties

The idea of grouping nodes is that all nodes in the same group are configured equally and, therefore, the settings of any group should be as equal as possible to the optimal setting of each node in that group. Since the optimal settings of a node are not directly observable, a model has to be used, which allows inferences on the optimal settings and the its input has to be observable.

Since the optimal settings of a node are related to the node properties, these properties can act as a model. Grouping those nodes with same properties results in groups of nodes where

all nodes, belonging to the same group, have similar optimal settings. Therefore the group formation component has to group nodes with equal properties.

Algorithm Transparency

When developing adaptive algorithms, the developer has to name the configurable parameters. Based on these parameters, the best performing settings for a specific network situation can be evaluated using simulation runs. Introducing groups results in the problem that the group formation component has to know the network properties which are relevant for algorithm parameters. In other words, if the groups should be used to adapt a routing protocol with the tunable parameter *radio strength* and *maximum retransmits*, then the group formation component has to know which network properties should be used for grouping. In the routing example the number of neighbors influences the behavior of the routing algorithm and, therefore, should be used, but the brightness of the environment does not.

The optimal behavior would be if the group formation component could identify the relevant properties itself and hence be transparent to the used algorithms. Without this feature, the system cannot select a model for group formation itself and, therefore, the model selection have to be performed explicitly by the algorithm developer.

Number of Groups vs Group Size

The main reason for introducing groups is the fact that it is infeasible to configure each node individually. The problem does not concern the adaptation of the nodes or the lookup of the setting to use, in fact the problem is to initially determine the configuration for each node to use. The investigation of optimal configurations requires to simulate the network and search for the optimal setting in the parameter space. Since the space of settings grows exponentially with the number of nodes, such an approach does not scale. Using groups decreases the parameter space, but the space still grows exponentially with the number of groups. The effort of finding optimal settings for groups can be decreased by simulating the behavior of each group individually. Every group of nodes can be understood as a box with links to other groups, with specific traffic patterns.

Individual simulations have the drawback that relations between settings of different groups cannot be identified. This behavior could result in the situation that two neighboring groups configure their settings in such a way, that no communication between the two groups is possible. Another problem is that such a simulation only has a local view of the network and, therefore, global optimizations are difficult or even impossible.

Independent of the used simulation method, the effort of the settings investigation mainly depends on the number of groups. While discussing the effort of finding the optimal settings other issues based on number of groups must not be forgotten. The first issue is that a higher number of groups results in potentially better results, since there are more possible configurations, and the second issue is that larger group sizes result in more effort at runtime, when the cluster head has to collect the state of nodes in the group.

Due to the pros and cons of group size and number of groups, the group formation models have to strike a balance between these.

Stable Groups

Reconfiguration of nodes requires communication effort to send the current settings to every node and to synchronize to ensure that every node runs the new configuration. Additionally, the reconfigured components may require a reinitialization. During the reconfiguration time, the application is stopped or works with limitations and, therefore, the groups should be stable and not change in consequence to small environment changes.

Chapter 3

Related Work

This section introduces related work to this thesis. The focus of this work is to develop models to form groups for adaptation purposes. To the best of our knowledge, no such work has done previously. Therefore, in following subsections projects working on related topics are discussed.

3.1 Adaptive middleware

When adapting algorithms in sensor networks, a framework is required, which covers tasks like code distribution, network reconfiguration and the determination of optimal configurations. This section gives an overview on projects developing adaptive middleware systems.

The first member of the category containing adaptive middleware system is *TinyCubus* [MLM⁺05]. In general, *TinyCubus* consists of three major building blocks: *Cross Layer Framework*, *Configuration Engine* and *Data Management Framework*. The cross layer framework allows information sharing between different layers of the software stack without building a deeply coupled application. Exchanging software components at runtime is required to adapt a deployed sensor network to new demands. The configuration engine addresses this functionality. Since every application has different quality of service requirements on the network, the data management framework allows an automatic adaptation of system parameters to fulfill these requirements. A detailed discussion of this middleware is provided in Chapter 2.

Impala [LM03] is another middleware focusing on the adaptation aspect. To reach this goal Impala allows to build modular, event driven and plugin based applications. A typical application using this middleware consists of several algorithms with equal functionality. Using the *Application Adapter* it is possible to select the optimal algorithm for a specific situation and its quality of service requirements. In the case when no optimal algorithm is installed on the sensor node, it is possible to install new modules using the *Application Updater*. Due to the long runtime of sensor networks, robustness of application is important. To ease the development of application, Impala introduces an event based programming model. The *Event Filter* receives incoming events and forwards them to the application module to handle them. The adaptation itself is controlled by the Adaptation Finite State Machine (AFSM). This machine allows the developer to specify the behavior of the system in case of node, network, or environment changes.

The guiding idea behind Milan (Middleware Linking Applications and Networks) [HMCP04] is the existence of several ways to get the same information. The differences between these information sources are concerning the quality of service. Quality of service means that the amount of required energy or the accuracy of the information differs. Since every software component needs a specific minimal quality of service, Milan calculates and configures the networks to use the optimal information sources.

3.2 Clustering

Another related topic is the area of clustering algorithms. Combining nodes to clusters introduces a hierarchical structure to the network which allows to implement localized algorithms. One benefit of such algorithms is that data of several nodes can be processed in the cluster, which reduces the amount of data that has to be sent through the network. This section discusses different approaches for clustering sensor networks.

Due to the fact that communication in sensor networks is very energy consuming and all inter-cluster communication makes use of the cluster head, this node should be equipped with extra energy or the role of the cluster head should circle in the cluster. Clustering sensor networks is a common design pattern in sensor networks and, therefore, several solutions for circling cluster heads exist [YF04].

Other factors influencing the group formation are application reasons, for example, when several nodes are required to retrieve events of the environment. Explained on a concrete example, in the *Sustainable Bridges* application [MSKG05] several nodes are used to monitor the structure of the bridge. On the occurrence of a vibration the nodes can calculate the origin of the vibration, by exchanging their monitored data and using triangulation. Since the data used for location determination is much higher than reporting the occurrence to the base station, it becomes obvious to optimize the communication between the nodes. In other words, the group of nodes used to detect and locate vibrations can be understood as a distributed sensor.

Using clustering algorithms allows efficient data gathering in sensor networks [DKN03]. Based on the formed clusters, it is possible to perform in-network aggregation to reduce the amount of data that has to be routed through the network. An inherent property of a sensor network is the existence of several nodes able to collect the required information. The authors [DKN03] are describing a heuristic method to select the data sources and maximize the lifetime of the system.

HEED (Hybrid Energy-Efficient Distributed clustering) [YF04] is a protocol for cluster formation. Every cluster contains one cluster head which is able to communicate with every node in the cluster directly. All data generated in a cluster is routed to the base station using the cluster heads. HEED considers several conditions when selecting the cluster heads. Since the primary goal is to extend the network lifetime, the remaining energy of a node is crucial.

3.3 Routing Algorithms

The third component needed to adapt algorithms in sensor network are the algorithms itself. Since practically every sensor network is used to retrieve data about the environment, the collected data has to be transferred to the base station, which is connected to the end user. Due to the limited transmission range of the sensor nodes, a direct communication with the base station is often impossible and, therefore, multihop communication is used. Routing algorithms are used to discover such a path of nodes between the data generator and the base station. Since the routing algorithm is a fundamental component of most sensor networks, the adaptation of such algorithms is the target of the developed group models. This section deals with the topic of routing algorithms.

Routing data in sensor networks has many different faces. Depending on the node mobility completely different protocols have to be used. In scenarios without or with low mobility tree based routing protocols [JMC⁺01] are commonly used, because of the low overhead, since every packet is routed on exactly one way to the target. Routing using this approach is split in two phases. The first phase is used to create the routing tree, by selecting the neighbor with minimal distance to the base station as parent node. It is now possible to send data to the base station by recursively forwarding packets to the parent node.

Another approach to route data through the network is based on location knowledge of nodes. The foundation of geographic routing [BCSW98] is the idea, that data is sent to the neighbor nodes with smaller distances to the target. Due to the mobility of nodes the exact position of the target node is not known to the sender. In case of the DREAM (Distance Routing Effect Algorithm for Mobility) protocol the sender calculates the area, based on the last known location and the maximum node speed, in which the target is located. The data is then sent to all neighbor nodes in the cone, defined by the target area and the source node, which again calculates the cone and forwards the data. In scenarios where the mobility is too high or location update happens too infrequent, the cone increases in size and, therefore, the geographic routing results more or less in flooding of data.

For medium or high mobility, routing algorithms are used which create the routing tables on demand. AODV (Ad-hoc On-Demand Distance Vector Routing) [Per97] is a reactive routing protocol for ad-hoc network. Reactive protocols are based on the idea that network topology changes frequently. Due to this changes, the creation of routing tables for all possible routes requires too much energy. Since no routing table exists, the routing of every packet requires a path discovery. The effort to find this path is reduced by introducing caches. Due to the mobility of the nodes, the discovered route is only cached for a short period, and the sending of later frames, even to the same target, requires a new route discovery.

OLSR (Optimized Link State Routing Protocol for Ad Hoc Networks) [JMC⁺01] provides an extension to the link state routing protocol. The classic link state routing is based on exchanging link state packets covering the 1-hop neighborhood of a node. Combining the neighborhood of every node results in a global view of the network, which enables the calculation of the shortest paths to every target. To reduce the amount of information which has to be flooded through the network, OLSR publishes only a subset of a node's neighborhood. In the original link state protocol, every node is used to flood the link state packets, which introduces duplicated messages and, therefore, large traffic. In OLSR every node de-

termines a subset of its neighborhood to flood its packets. Based on the smaller packets and the smaller number of nodes, OLSR is able to reduce the required traffic to distribute the global view.

AODV and OLSR have to estimate the costs of a path. The common used metrics is the usage of the number of hops. Alternative metrics based on the power consumption are provided by [SWR98]. The metrics have the goal to ensure that the remaining energy of all nodes is equal, or to maximize the time until the network is partitioned, as a consequence of nodes without remaining energy.

3.4 Monitoring Frameworks

The authors of [ZGE02, ZGE03] are developing a framework to transfer the state of a node to the base station. The idea behind these frameworks is to include state information in messages that have to be sent anyway. Using this piggybacking no additional messages have to be sent and, therefore, the overhead of the state collision is minimal. In addition, it is possible to use in-network aggregation to abstract from the individual nodes and transfer only the average value of a region to reduce the amount of data to be send beyond this amount.

Chapter 4

Sensor Network Properties

Wireless sensor networks have several application areas which require different sensor hardware (see Figure 4.1) and software. In some application scenarios, the sensor data must be routed continuously to a base station. These scenarios are for example the observation of temperature or humidity of a forest. This continuously data flow requires an efficient routing topology with minimum packet overhead to increase network lifetime.

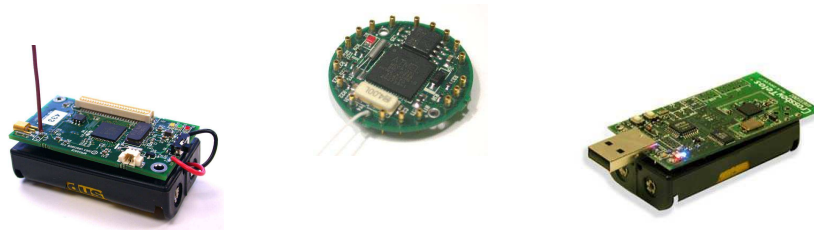


Figure 4.1: Sensor nodes: Crossbow Mica2 [Cro06a], Mica2Dot [Cro06b] and TelosB [Cro06c]

In other scenarios like intrusion detection systems [RZL06], the observed events must be sent reliable to a base station. Due to the fact that these events do not occur very often, the focus on the routing system is more on reliability than on energy consumption.

In both scenarios discussed above, the topology of the sensor nodes is static. In contrast to these scenarios, there are other application domains where sensor nodes are highly mobile. An example of such an application domain is a driver assistant system [MHD⁺03]. This system monitors the current traffic situation and warns the driver of the car of traffic jams. The sensors are connected to the power supply system of the car and, therefore, energy consumption is not critical in such a system. Due to the mobility, the density of cars and, because of this, the density of the sensor nodes is continuously changing.

Summing up the short introduction of possible application domains, it can be stated that there are many characteristics that have to be taken into account when trying to group nodes in such scenarios. The characteristics can roughly be classified in three categories:

- **Node Characteristics** [Section 4.1], describing the hardware of a single sensor node.
- **Network Topologies** [Section 4.2], describing the possible topologies formed by sensor nodes.
- **Application Characteristics** [Section 4.3], describing properties of applications that are relevant grouping nodes.

In the following sections, these three categories are further subdivided and discussed in detail. The discussion of the network and application characteristics includes the question, whether a specific property can be used for group formation.

4.1 Node Characteristics

This section identifies those characteristics of a sensor network that are based on individual nodes. One of the major goals of a sensor network is to collect information about the environment and, therefore, a sensor node needs technical equipment to get those information. This equipment is typically a simple sensor to get physical properties of the environment or of the node itself. These properties can be for example the temperature or the position of the node.

In productive systems, often not the individual sensor readings, but the combination of several readings is interesting. When the application is to track animals moving through a park, it is not relevant if sensor s_i saw the animal and the position of that sensor is unknown. Another point is that there are often several sensor nodes detecting the animal and, therefore, the application has to combine the events of those sensor nodes to get the exact position. To allow this combination the sensor nodes need data processing hardware to handle and transform those events.

At last, the collected information has to be transmitted to a base station to be available for usage. This communication requires hardware which contains a sender and a receiver.

In the following sections the three node components and other hardware devices are further discussed.

4.1.1 Sensing Hardware

As mentioned above, a sensor node has several sensing hardware to collect information of the environment. These sensors can be divided into sensors getting information about the environment and those getting information about the node itself.

Sensors belonging to the first category are directly used by the application or, in other words, the reason for deploying the sensor network is the gathering of those sensor readings. The second category contains those sensors that are not in the focus of the user. The readings acquired from those sensors are used by the application to fulfill the requirements of the user.

An example set of sensors is listed in the following:

- Collecting environment properties:
 - temperature
 - brightness
 - humidity
 - motion detection (other objects)
 - position (GPS)
 - vibration
- Collecting node properties:
 - battery charge
 - received signal strength indication (*RSSI*)
 - number of neighbors
 - memory state (installed programs)

The node properties *number of neighbors* and *memory state* are based on virtual sensors, because they are acquired by software components and, therefore, require no real hardware devices.

4.1.2 Data Processing Hardware

To allow transformation or aggregation of sensed data, a sensor node is equipped with data processing hardware. This hardware consists of a CPU with main memory for the computations and also secondary memory to store data. The CPU is 8-bit, low voltage and runs only with few MHz. The size of the main memory is about a dozen kilobytes.

The secondary memory has a size smaller than one megabyte and is non volatile, allowing to store currently not used programs or sensor readings. Table 4.1 provides an overview of common used hardware platforms for wireless sensor networks.

Mote Type	Mica 2	Mica 2 Dot	Telos	μ Part
CPU Name	ATmega128	ATmega128	T1 MSP430	12F675
Ram Size	4KB	4KB	10KB	64B
Flash Size	512KB	512KB	1024KB	1.4KB
Active Power (CPU)	33mW	8mW	3mW	-
Sleep Power (CPU)	75 μ W	75 μ W	15 μ W	-
Total Active Power	89mW	44mW	41mW	-

Table 4.1: Common sensor node hardware platforms [PSC05, TEC06]

The power to run a node comes from batteries which only have a limited amount of capacity. While Mica 2 and Telos Motes use two AAA batteries, the Mica 2 Dot requires a coin cell. Based on the capacities, listed in Table 4.2, a Telos node can operate about 70 hours in

active or 22 years in sleep mode (accounting only CPU sleep power). Even though sleeping all the time is not useful, the comparison of the two times shows, that switching the CPU in sleep mode saves a lot of energy. Another conclusion is that the energy, provided by batteries, is very scarce and, therefore, the limited energy has to be taken into account whenever designing applications for wireless sensor networks.

Battery	AAA	Coin Cell
Voltage	1,5V	3V
Capacity	1000 mAh	200mAh

Table 4.2: Battery based power supply

4.1.3 Communication Hardware

In sensor networks, the communication between nodes and from a node to the base station is based on wireless transmission. Using wireless instead of wired communication allows to deploy nodes easily, because no cables have to be installed. The absence of cables is a fundamental requirement in most scenarios of sensor networks.

The gained flexibility using this technology has to be paid with high energy costs for data transmission. The costs to transmit data are portioned to the sender and the receiver. The sender side costs are quite evident. The costs on the receiver side results from the fact that a receiver has to separate the wanted signal from background noise by amplifying the signal.

Based on the awareness that communication is very energy consuming for sender and receiver, there exist several counteractive measures to limit the energy wastage. The most obvious action is to avoid communication by processing the data at the source node [YG03] or to aggregate data to limit the amount of data to be transmitted [ABS03]. The processing of data using the CPUs of the sensor also consumes energy. When comparing the energy usage of transmission data and executing CPU instructions, see Table 4.3, it is possible to execute more than thousand instructions with the same energy consumption than sending one byte.

	CPU Instruction (MSP-430)	Radio TX (CC2420)	Radio RX (CC2420)
Energy ($\mu\text{J}/\text{byte}$)	0.0008	1.8	2.1
Ratio	1	2250	2600

Table 4.3: Comparison: computing data vs transmitting data [MDGS06]

Another action is to adjust the transmission power of the nodes' radio subsystem. Decreasing the sending power decreases the used power at sender side and, because of the lower transmission range, also the number of nodes receiving the data.

Another method to reduce the power consumption is to disable the radio subsystem at times where no incoming packets are expected. This approach is appropriate in scenarios where data is only sent in predefined intervals.

4.1.4 Other Hardware Devices

Location awareness of sensor nodes is required for many applications running on sensor networks. In scenarios like observation of animals, it is obvious that if a sensor reports the detection of an animal, the location of the sensor has to be known. When the sensor network is deployed node by node, then it is possible to configure every node with the exact position. In cases where nodes are deployed in a huge amount, by dropping them out of a plane or scenarios with moving nodes, it is impossible to configure every node with its position.

To get the required location information, sensor nodes can be equipped with GPS receivers [Xu03], which provide worldwide absolute position information. Since GPS is based on satellites, it cannot be used in buildings. Another drawback is the demand for additional hardware which requires larger nodes and consumes energy.

To conquer the drawbacks a software solution was developed. Based on triangular methods [PIP⁺03], it is possible to get the relative position of nodes. In cases where some nodes or the base station is equipped with a GPS device, it is possible to calculate the absolute position of any node in the network.

4.2 Network Topologies

The topology formed by the nodes of the sensor network depends on the requirements of the application. In some applications the position of nodes is given by the objects to monitor, because the nodes are attached to the objects while other applications allow the designer of an application to position nodes freely.

Other characteristics of the network topology are implicated in the fact that some applications need moving nodes. As mentioned above, the nodes are in some applications attached to a monitored object and, therefore, the nodes are moving with the object. A sample application for this sort of mobility are nodes integrated in a car.

In the following section, the characteristics of the formed network are discussed in detail.

4.2.1 Regular vs Random Structures

The structure of a sensor network can be a regular, which means that the nodes are deployed in a formation based on geometric forms. When monitoring the growing of crop, see Figure 4.2, it suggests itself to deploy the nodes in rectangle with the size of the acre to observe.

In other scenarios, for example when a sensor network is used to get information on the propagation of forest fires, see Figure 4.3, a huge amount of nodes has to be deployed. The deployment of so many nodes by hand is not feasible and, therefore, the nodes are distributed using planes, dropping out a large amount of sensor nodes. The position of an



Figure 4.2: Structured Node Deployment: Monitoring crop growing

individual is random, however, depending on the dropping strategies, there exist patterns in the structure of the nodes, e.g. the nodes are arranged in circles around the dropping zones, with many nodes in the center and few nodes at the edges.

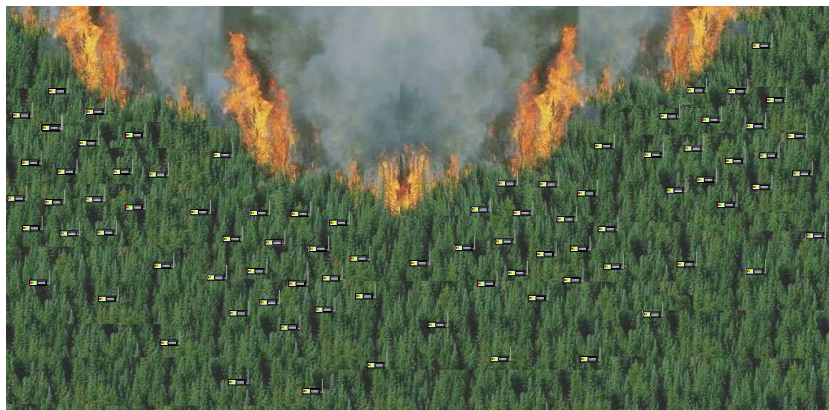


Figure 4.3: Random Node Deployment: Monitoring forest fire

The structure of the network primarily depends on the deployment of the nodes. Networks without uniformly distributed nodes allow to use the arrangement of nodes to form groups for adaptation. A possible group formation pattern is, for example, to group nodes of each dropping zone. Since the nodes are not equally distributed in each dropping zone, the density, discussed in Section 4.2.4, of nodes varies in each zone.

In contrast to static nodes, the structure of the network can change over time when nodes are moving. In the next section, the characteristics depending on mobility are discussed.

4.2.2 Static vs Mobile Topologies

In scenarios, where nodes are moving, the topology of the network is changing all the time. Several implementations of applications require knowledge of the node's neighborhood. The neighborhood of a node can be used to build routing tables or to aggregate data. With moving nodes, the neighborhood of nodes changes all the time and requires effort to keep them up to date.

In general, the mobility of nodes can be divided in *random mobility* and *regular mobility*. To the category of *random mobility* belong those scenarios, where all nodes are moving individually, while in the *regular mobility* the movement of nodes is restricted by the environment. Due to the restricted mobility, in *regular mobility*, see Figure 4.4, the relative speed between nodes in the same group moving in a specific direction is much lower than the speed of the group.

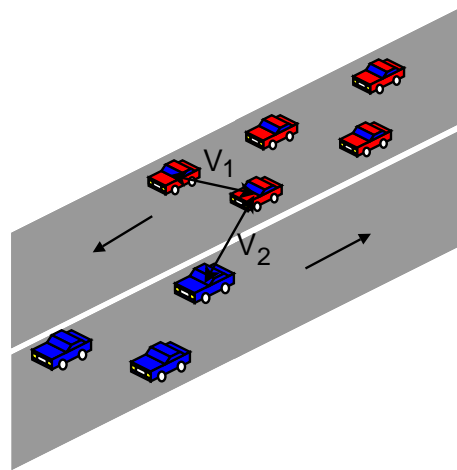


Figure 4.4: Moving Groups: speed V_1 is much smaller than speed V_2

A fundamental problem, arising with the mobility of nodes, is the short connection time, see Figure 4.5, between nodes. When the lanes of two nodes cross, then depending on the speed of the nodes, there exists a short period where data exchange is possible.

Static topologies ease applications to form clusters of nodes to aggregate sensor readings or to implement energy aware routing protocols. In case of adaptation, the mobility of nodes results in ongoing changes of network's properties and, therefore, reconfigurations of the network, which can, in worst case, erase the benefits of the adaptation. Another problem is, assuming groups are connected, that nodes are joining and leaving groups all the time. Connected means that every pair of nodes, belonging to the same group, can communicate using only other nodes of the same group as repeaters. To conquer the ongoing group joining and leaving, the group models have to cover mobility explicitly.

4.2.3 Free vs Constrained Topologies

When deploying a sensor network, the nodes form a graph, where sensors are nodes and radio connections between nodes are edges. In scenarios like monitoring the growing of crop,

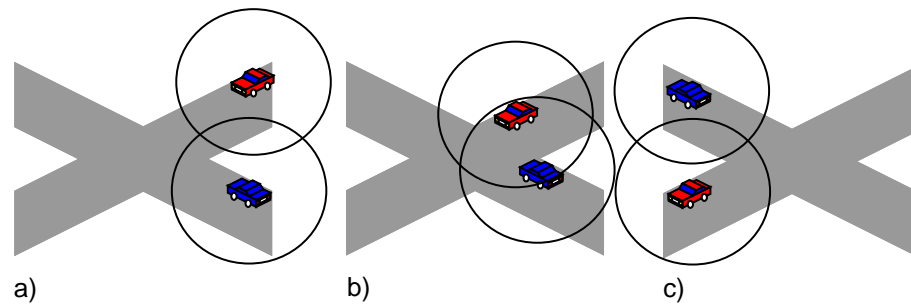


Figure 4.5: Connection time between moving nodes: No transmission possible in a) and c), b) allows data exchange

see Figure 4.2, the connection between nodes is primary dependent on the sending power. Increasing the sending power results in larger neighborhoods. In general, the influence of the environment is limited and data can be routed on many ways to the base station.

In other scenarios, like Sense-R-Us [MMLR05], where a sensor network is used to build a smart research environment, and nodes are placed in office rooms, meeting rooms and corridors as well as attached to employees, to get information about the position of employees, the environment prunes the connection graph. Assuming that sensors can communicate through doors, but not through walls, the connection of nodes is limited to nodes in the same room or between nodes of connected rooms. In contrast to the previous example, the inference of the environment is high and data has to be routed according to buildings layout.

Applied to grouping formation, the constraints of the environment can be used to group those nodes in the same room. Using this group formation model, it is possible to adapt the parameters of nodes in different rooms individually and optimize the communication between rooms. Based on these groups, it is possible to have different configurations for nodes in empty office rooms and meeting rooms with a meeting in progress.

4.2.4 Density of Nodes

A further characteristic of the network topology is the density of nodes. In network topologies with a high density, the nodes are interconnected with many links. The high interconnection allows the routing protocols to find many different routes to the base station and, therefore, the overlay network formed by those protocols is very robust.

A high density of nodes has also negative implications. With an increasing density, the probability of collisions increases as well. The number of irrelevant messages received by a node also rises with higher density of nodes and, therefore, the required energy also rises.

Using adjustable transmitters, introduced in Section 4.1.3, the sending power can be decreased and the number of neighbors limited. Smaller sending power limits the number of collisions and the number of irrelevant received messages.

Summing up this section, in scenarios where the density of nodes varies in the network, this network property can be used for group formation. During the adaptation process it is possible to adjust the transmitters to adapt the density to the network situation. The adaptation density can be used to improve the behavior of algorithms used in the sensor network.

4.2.5 Number of Base Stations

The main objective of a sensor network is to collect information about the environment for the user. This information has to be routed to the base station and from there to the user's computer. In typical scenarios, the amount of data a node has to forward, increases with the closeness to the base station, because all data passes the few nodes around the base station. The amount of data, that has to be send, can be reduced, when data of different nodes is aggregated on the path to the base station, but the gradient in the amount of data to forward remains.

As described in previous sections the sending of data using radio transmitters costs a lot of energy and, therefore, installing multiple base stations interconnected with an infrastructure saves energy [Nea06]. The reduction of the required energy results from the smaller average distance to the base station and, therefore, the lower number of hops. Another reason is the distribution of the traffic to many nodes, which increases the lifetime of the network.

Sensor networks with several base stations can be understood as several independent networks by assigning each node to its closest base station. It is now possible to adapt each network for its own. Based on this interpretation the nodes can be grouped depending on the closest base station.

4.2.6 Homogeneous vs Heterogeneous Networks

The topology of sensor networks is often similar to peer-to-peer systems where all nodes have the same functions. All nodes collect data using their sensors and route that data to the base station using multihop communication. With increasing network size, the gradient in the amount of data, mentioned in Section 4.2.5, results in a short lifetime of the nodes around a base station.

An approach to conquer this problem is to install heterogeneous nodes [MR04]. One part of the nodes is responsible for collecting sensor data, and the other part is used to forward the data. One forwarding node and several sensing nodes are grouped to a cluster. All data is transmitted to the forwarding node using low transmission power to increase the lifetime of the sensing nodes. The forwarding nodes are equipped with a larger battery pack, which allows the usage of higher sending power.

4.3 Application Characteristics

While the last sections described the hardware and the topology of a sensor network, this section focuses on the software layer implementing the application logic. As discussed in the previous section, a major problem in sensor networks is the routing of data from the sensors to the base station. Therefore, Section 4.3.1 discusses this topic in detail.

The purpose of the thesis is to form groups in sensor networks for adaptation. There are several scenarios, where groups based on application knowledge are already defined. In Section 4.3.2 these groups are reviewed.

4.3.1 Network Traffic

Using tree based protocols with more or less static trees has the benefit that the tree can be reused for aggregation of data. When the user is interested in the average value of a set of sensors, only a cumulated value has to be routed through the network which conserves a lot of energy.

When comparing the different routing approaches, it becomes obvious, that the amount of data sent by each node varies and, therefore, the traffic differs in the network. Due to the fact that the target node for most data is the base station, the network traffic increases with the closeness to the base station. Based on this phenomenon, the distances to the base station can be deduced from traffic of a node.

The traffic consists of several components, which allow different conclusions about the position of the nodes as well as their role in the network. The traffic components and possible conclusions are:

- Sent data (TX): Number of nodes that are lower in the routing tree
- Received data, addressed to this node (RX_{node}): Number of nodes that are lower in the routing tree
- Received data, overheard from other nodes (RX_{other}): Distance to the base station

TX and RX_{node} allows to estimate the number of children in the routing tree, because the traffic of all children has to be received and forwarded to the base station. To determine the exact number of children the knowledge of the amount of traffic generated by each node is required, but even without that knowledge it is possible to estimate the number by using a virtual fix amount. If every node uses the same virtual amount, it is possible to compare the number of children which is sufficient to form group of nodes having similar number of children.

In contrast to RX_{other} the first two components cannot be used to estimate the distance to the base station, because two nodes with no children in the routing tree have the same TX and RX_{node} independently of the distance to the base station. In case of using RX_{other} , a node near the base station has a neighboring node with a large number of children in all probability. Since this neighboring node forwards a lot of data, the node without children also overhears the traffic and, therefore, the distance to the base station can be assumed to

be small. Nodes with a large distance and also no children will not overhear the forwarded traffic of many nodes and, because of this, their distance can be assumed to be large.

When using the traffic to form groups for adapting routing protocols, the usage of TX and RX_{node} should be avoided, since they are closely related to the actual routing tree. The problem which results from using those components is the occurrence of feedback loops. The actual routing tree affects the creation of the routing tree.

Principally RX_{other} is also affected by the actual routing tree, but small changes in the tree do not change the RX_{other} significantly. This circumstance becomes clear when considering a node n_1 which has two nodes p_1, p_2 as potential parents. Both nodes p_1, p_2 will receive the data, n_1 sent to the selected parent and, therefore, RX_{other} is not affected by this change on the routing tree.

4.3.2 Application Based Groups

Some applications require the existence of nodes with different sensors. Depending on the sensors, the amount of data generated by the sensors can vary a lot, which enables optimizing options how to configure the radio system or the routing protocols. In such scenarios it makes sense to group nodes with equal sensors for adaptation.

There are also applications where sensor readings from an individual node are not relevant, in fact the combination of several sensor nodes provides the required information. In comparison to the required information a lot of information has to be exchanged between the involved nodes. Taking the different amounts of data into account, grouping nodes with the same properties and adapting their settings can be useful.

4.4 Summary

After characterizing the sensor network in general, the different properties are investigated for group formation usage. Summing up the different properties it can be stated that the structure of a network can be used for group formation. The structure can result from the deployment of nodes or can be based on application knowledge. The different expected traffic in parts of the network, resulting from the distribution of density of nodes or from different used sensors, has an impact on the group formation, too.

Based on the requirements, resulting from the adaptation of *TinyCubus*, discussed in Section 2.2, and the characteristics derived from sensor network properties, the next part introduces different group models.

Part II

Group Formation Models and Theoretical Discussion

Chapter 5

Model Basics

This chapter covers the basics used by the models introduced in Chapter 6. Therefore, the assumption on the network, fundamental algorithms and optimal group properties are discussed.

5.1 Network Assumptions

When developing protocols for distributed applications, assumptions on the underlying network have to be made. These assumptions cover, for example, node failures. Some protocols assume that nodes are running all the time and do not fail. Other protocols also cover temporary failures of nodes. Based on these assumptions, different protocols for the same scenario have to be designed.

The development of group formation models for wireless sensor networks also makes assumptions on the network. Based on the properties of the deployed network, a model has to be selected, where all of its assumptions are satisfied. In the following sections assumptions made by the protocols are discussed.

5.1.1 Receiver Signal Strength Indication

Receiver signal strength indication (*RSSI*) is a metrics describing the signal strength at the receiving node, see Figure 5.1. The signal power radiated by the sender is attenuated by the medium or other obstacles between the sender and the receiver. The radio module of the receiver provides the *RSSI* value to software layer. In scenarios where every node sends with the same transmission power, based on that *RSSI* value the distance to the sender can be estimated. The *RSSI* values are, depending on the sensor platforms, arranged on the interval between 0 and 100, where 100 stands for optimal signal power.

5.1.2 Network Status Monitoring Frameworks

Centralized approaches for group formation require the base station to know the status of every node in the network. The status of a node in this context only covers those properties that are relevant for grouping nodes. Based on this requirement, the status of every node has to be transferred to the base station.

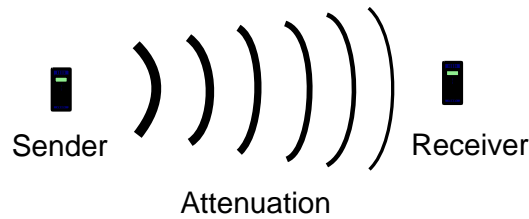


Figure 5.1: Radio signal propagation

Following the trivial attempt and send one message, covering all relevant node properties, from every node to the base station, is not feasible in praxis. Since every message has to be routed hop by hop to the base station, this approach results in $O(\sqrt{n} * n)$ messages (assuming the nodes are arranged in a square. In worst case $O(n^2)$ messages are required). In networks with hundreds or thousands of nodes this approach does not scale.

The discussion above only covers those messages that are required to transfer the state of the nodes, but does not solve the problem that the state of different nodes are taken at different times. Taking a consistent snapshot in a distributed system requires more effort or is even impossible depending on the required consistency level.

The centralized approaches, introduced in Section 6.4, do not require a snapshot with a hard consistency level. These models only require the average traffic rate or the average neighborhood of the nodes. The actual values at a specific time are rather destructive than constructive, because such a snapshot would cover all temporary sensing errors.

Based on the knowledge that average values and values from the past are sufficient, it is possible to use monitoring frameworks discussed in Section 3.4 to transfer the state of the nodes to the base station.

5.1.3 Mobile Nodes

Mobility of nodes changes the requirements on group formation models fundamentally because the topology of the network changes all the time. As discussed in Section 4.2.2, the connection time between nodes varies in the network. Based on the continuous changes on the topology, centralized approaches do not support mobility efficiently.

Since mobility requires extra effort to support topology changes, most models introduced in this chapter assume static nodes.

5.1.4 Bi-directional Links

In wired networks links between nodes are almost always bi-directional. Bi-directional means that if *Node1* can communicate with *Node2*, *Node2* can also communicate with *Node1* directly. In wireless networks links are not always bi-directional because different transmission powers of two neighboring nodes can result in uni-directional links.

Other factors effecting uni-directional links are based on the environment. Obstacles can scatter and reflect the signal which results in errors, uni-directional links or disallows communication. Assuming a connection between two nodes with a uni-directional link, it is coherent that the link has a poor quality.

Based on the discussed factors, models generally used in wireless networks have to assume uni-directional links. Taking into account that the design models are used to adapt the transmission power of the nodes, those models can assume equal transmission power settings during the group formation phase. Using the knowledge that uni-directional links have poor quality, it is possible to ignore them by considering only connections with a quality above a given threshold.

Summing up the discussion, it can be stated that in case of designing models for group formation, bi-directional links can be assumed.

5.2 Fundamental Algorithms

Different solutions for grouping nodes in sensor networks, discussed in Sections 6.2, 6.3 and 6.4, often use the same underlying algorithms to solve equal subproblems. Since there exist several possible solutions for those problems, this section gives an overview on them and discusses the pro and cons.

5.2.1 Graph Partitioning

Group formation in networks is, as the name suggests, closely related to a topic of theoretical computer science, named graph partitioning. Several models, discussed in the following sections, require the existence of solutions to solve graph partitioning. This section defines and provides an overview on that topic.

A graph is a tuple of two sets $G = (V, E)$, where V is a non-empty, finite set of vertices and E is a set of edges. An edge is a tuple (v_i, v_j) , where $v_i, v_j \in V$ represents a connection between vertex i and j . A subset of the directed graphs are undirected graphs, where following relation stands: $(v_i, v_j) \in V \leftrightarrow (v_j, v_i) \in V$. The definition of weighted graph extends a graph by a *weight function*: $w(e) \rightarrow n, e \in E, n > 0$. A graph is called *connected*, when there exists a sequence of links between all pair of vertices.

Figure 5.2 shows the weighted, undirected graph $G = (V, E, w)$ with $V = \{A, B, C, D, E, F\}$, $E = \{(A, B), (A, C), (B, C), (C, D), (C, E), (D, F), (E, F)\}$ and $w = \{(A, B) \rightarrow 1, (A, C) \rightarrow 2, (B, C) \rightarrow 3, (C, D) \rightarrow 1, (C, E) \rightarrow 1, (D, F) \rightarrow 2, (E, F) \rightarrow 2\}$.

A given graph can be separated in connected subgraphs G_i ($G_i = (V_i, E_i, w)$, where $V_i \subset V$, $E_i \subset E$, $\nexists v : v \in V_i$ and $v \in V_j$ with $i \neq j$ and $\nexists e : e \in E_i$ and $e \in E_j$ with $i \neq j$) in two different ways: removing nodes or removing edges. In context of this thesis the removal of edges is used, to separate the graph and, therefore, the removal of nodes is not discussed here. The sum of weights of the removed edges is called the cost of the separation. Since the weight of the edges stands for the quality of a link, the cost function has to be minimized. Another parameter is the size s of created subgraphs. In addition to the size a deviation ε is given, which allows subgraphs with $s \pm \varepsilon$ vertices.

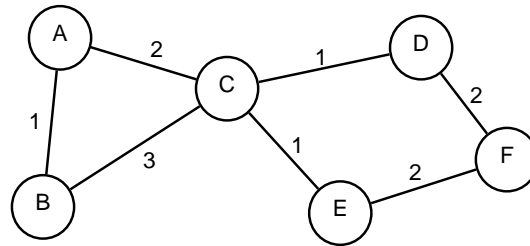


Figure 5.2: Weighted, undirected graph

The problem of graph partitioning is to find for a given graph $G = (V, E)$, a size s and a tolerance ε an optimal set of edges for removal, which results in subgraphs with a number of vertices $s \pm \varepsilon$. The Algorithm 5.1 computes the optimal set of edges, whose removal results in right sized subgraphs with minimal costs. Since the algorithm calculates all possible separations of the graph and checks, whether the size of those is in the given limit, the algorithm has an asymptotic runtime of $O(2^n)$. Graph partitioning is a NP-hard problem and, therefore, no efficient (runtime lower than $O(2^n)$) algorithm is known today.

Data: Graph $G = (V, E)$, Size s and Deviation ε

Result: Set of edges $edges_for_removal$

```

1  $edges\_for\_removal = \{\}$ ;
2 foreach combination of edges  $E'$  do
3    $G' \leftarrow (V, E \setminus E')$ ;
4   foreach connected subgraph  $g$  in  $G'$  do
5     if not( $abs(number\_of\_vertices(g) - size) \leq \varepsilon$ ) then
6       break;
7     end
8     if  $costs(edges\_for\_removal) > costs(E')$  then
9        $edges\_for\_removal = E'$ ;
10    end
11  end
12 end
13 return  $edges\_for\_removal$ ;

```

Algorithm 5.1: Graph Partitioning

When applying Algorithm 5.1 to graph G (Figure 5.2) with $size = 3$ and $\varepsilon = 0$, the two edges (C, D) and (C, E) will be removed. Figure 5.3 shows the two resulting subgraphs.

In many situations, like for partitioning a sensor network, there is no strict demand for the optimal solution. A nearly optimal separation of the networks is enough and, therefore, algorithms discussed in [Els97] can be used to approximate the optimal separation. Since no optimal solution is calculated, the runtime is much lower than the theoretical limit to search the optimal solution. Since graph theory is beyond the scope of this thesis, the algorithms are not discussed here in detail.

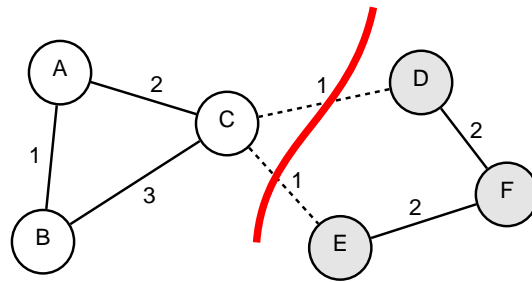


Figure 5.3: Separated graph

5.2.2 Cluster Head Election

After assigning every node to a group, a cluster head for every group has to be elected. When using a centralized approach the election is trivial, because the centralized component knows every member of a group and, therefore, can select for example the node with the largest ID.

In distributed approaches the election cannot be performed in that way, because no node knows initially every member of its group. Since the election of a cluster head has to be performed in many models, a distributed cluster head election algorithm is introduced here.

The election is divided in three phases [GM82]:

- **Explosion Phase:** Election is announced to the leafs
- **Contraction Phase:** Cluster head is searched by propagation of the largest ID to the center
- **Information Phase:** Nodes are informed about new cluster heads

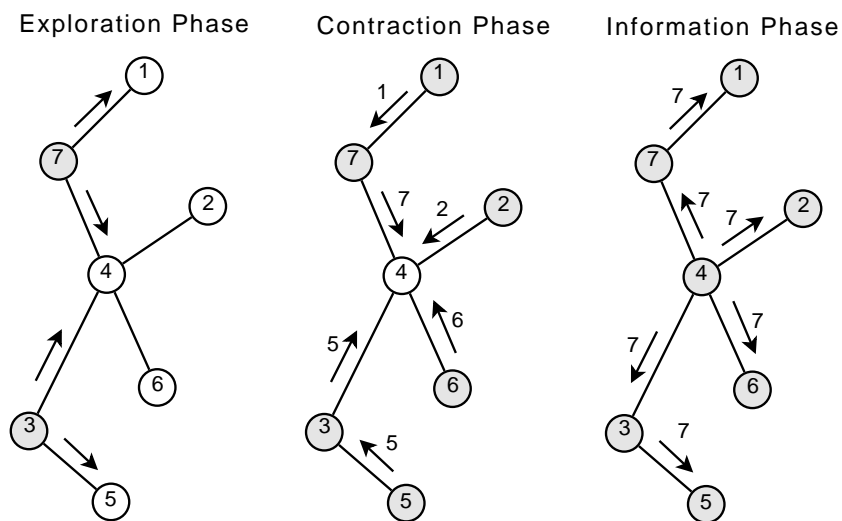


Figure 5.4: Cluster Head Election

During the *Explosion Phase* one or more nodes broadcast an election announcement. The messages are only forwarded between nodes of the same group. When the announcement arrives at a leaf the *Contraction Phase* starts. The announcement is send back containing the node ID of the leaf. Every node receiving such a message replaces the ID in the message, if the own ID is larger. Nodes receiving a message on all links, know the new cluster head and, therefore, start the *Information Phase* by flooding the ID of the cluster head. Now every node knows the cluster head. The message flows for cluster head selection are visualized in Figure 5.4.

Since communication in sensor networks costs a lot of energy, the messages used to elect a cluster head have to be taken into account when discussing the costs of a group formation algorithm. The costs using the distributed cluster head election algorithm are listed in Table 5.1.

Phases	Number of messages
Explosion	n-1 (one message on every edge)
Contraction	n-1 (one message on every edge)
Information	n-1 (one message on every edge)
Total	3n-3 (every node has to send 3 messages)

Table 5.1: Required messages for electing a cluster head

5.2.3 Neighborhood Value

The neighborhood of a node is a common used criteria in various models to form groups. Nodes with similar neighborhoods are grouped. The problem following this approach is the comparison between two neighborhoods and the description of the differences between them. To conquer that problem a metrics has to be developed, which allows that comparison.

During the formation of groups, a large number of neighborhoods has to be compared. Since the detection of a node's neighborhood is provided locally by every node, it is necessary to send the neighborhood through the network. To limit the size of data to be sent, the neighborhood has to be mapped to a single value (*Neighborhood Value*). This value can be exchanged and compared efficiently.

In addition to the size of the value, other requirements exists. The *Neighborhood Value* should cover the number of neighbors, because the behavior of many algorithms is dependent on the neighborhood size. Another important factor is the distance between a node and nodes in its neighborhood. In case of routing algorithms distant nodes are not that important, because long distance communication is very energy consuming. Related to the previous requirements is the demand that the *Neighborhood Value* should reflect the density of a node.

To ease the usage of the *Neighborhood Value*, the domain of the value should be limited, which means that there exists a minimum and a maximum value. Based on the upper and lower limit it is possible to scale the value to the discrete interval between 0 and 255, which allows efficient handling on sensor nodes.

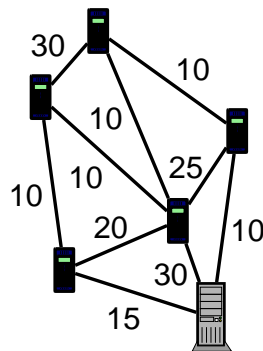


Figure 5.5: Sample Network

Figure 5.5 shows a small sensor network with five nodes and a base station. In addition to the hardware components the links between them and the quality of the links are visualized. Based on that network different approaches to model the *Neighborhood Value* are introduced.

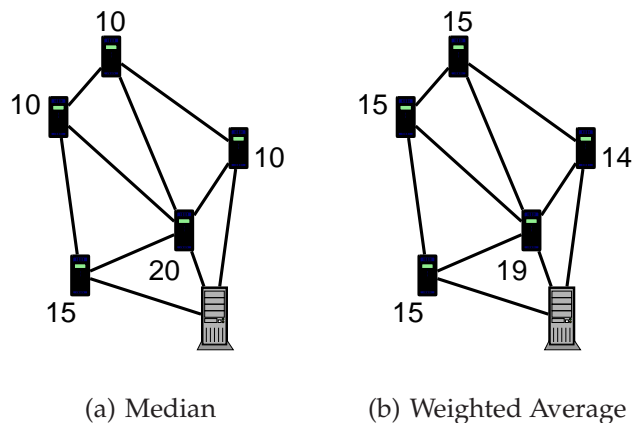


Figure 5.6: Possible *Neighborhood Value* metrics

The first approach modeling the *Neighborhood Value* is to use the median of all qualities of links, visualized in Figure 5.6 (a). This approach has the benefit that outliers are not covered and, because of this, very near and far away nodes are ignored. Since link qualities have a limited domain, the median also has that feature. The drawback of that approach is the fact that a given value is not influenced by the neighborhood size and, therefore, does not allow to make assumptions on the density.

Figure 5.6 (b) shows the *Neighborhood Values* based on the *Weighted Average* approach. The basic idea behind this approach is to calculate the average link quality of all links to nodes in the neighborhood. Since near and far nodes are assumed to be outliers, these qualities are weighted with a low factor. Nodes near the median are weighted with a high factor. The

factors used in Figure 5.6 (b) are 1 for the lowest and highest link, 2 for the second lowest and highest link and so on. In general, this calculation has the same benefits and drawbacks as the *Median* approach.

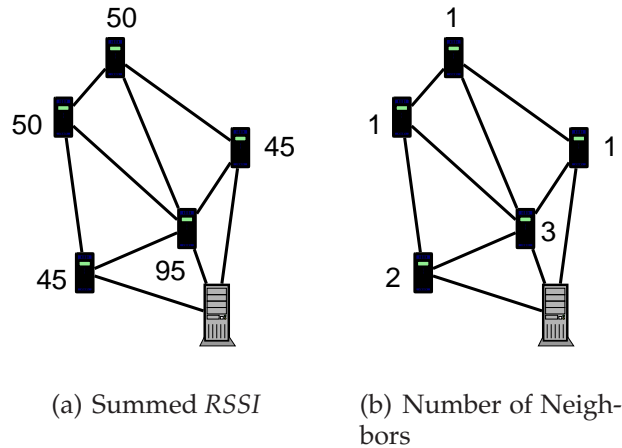


Figure 5.7: Possible *Neighborhood Value* metrics

Another approach, see Figure 5.7 (a), adds up all link qualities. The *Neighborhood Value* covers the number of neighbors and the distances between nodes. Even though the density is only related to two values this approach nevertheless does not model the density. The problem is that many distant nodes result in the same value as some near nodes. Another drawback is the domain of the values calculated by this approach. It is not possible to define an upper limit, unless the total number of nodes in the network is known. Knowing that number the upper limit can be defined as the product of node count and maximum link quality.

The last introduced approach, see Figure 5.7 (b), is based on the idea that density can be understood as the number of nodes within a given radius. Following this idea the value is calculated by counting those nodes having a link quality above a given threshold. Using this approach the number of neighbors and the distances are covered as well as the density. In general, it is impossible to define an upper limit, but following the argument discussed in the previous model description, the number of nodes can be used as the upper limit. Based on application domain knowledge lower limits can also be used, because the protocols do not change their behavior with increasing neighbors at a specific point.

Requirements	Median	Weighted \emptyset	Σ RSSI	# Neighbors
Number of Neighbors	○	○	✓	✓
Node Distances	✓	✓	✓	○/✓
Density	○	○	○	✓
Limited Domain	✓	✓	○/✓	○/✓

Table 5.2: Fulfilled Neighborhood Value Requirements

Table 5.2 summarizes the features of the discussed models. Since the approach, counting the number of neighbors above a given threshold, promise the best suitability for usage in group formation, this approach is used in the following sections for modeling the node neighborhood.

5.3 Optimal Group Properties

Groups for adapting routing algorithms have to fulfill specific requirements. These optimal group properties, discussed in this section, extend the requirements resulting from *Tiny-Cubus* which are discussed in Section 2.2.

5.3.1 Density Consideration

Tuning the transmission power of nodes allows to optimize the functionality of routing algorithms. Low power values reduce the probability of collisions and save energy. High power values allow to communicate over large distances and result in a highly connected network.

The optimal transmission power is closely related to the density of nodes. A node with a large neighborhood is able to communicate with many nodes with a low transmission power. Due to this relationship an optimal group model has to take the density of nodes into account.

5.3.2 Traffic Consideration

Sending data in sensor networks costs a lot of energy. Based on this fact, losing data caused by collisions has to be avoided. Since the number of collisions is related to the amount of traffic in a region, nodes with equal traffic patterns should be grouped. Using those groups it is possible to configure nodes for minimizing the number of collisions.

5.3.3 Group Borders

Adjoining groups are normally configured with different settings, because in the other case, these two groups can be merged. Nodes near the border between these two groups are more or less randomly assigned to one of these groups, because slight property changes would result in joining the other group. Node a in Figure 5.8 would become a member of the left group in case of a small position change.

The effect of these small changes is amplified by using groups with borders shown in Figure 5.9 (a). The problem of borders in this example is that there exists only a small common border between both groups. The average distance between a pair of nodes belonging to different groups is high and, therefore, the properties of both groups are supposable different which results in different configurations for both groups. Such a configuration results in a good performance at most nodes of both groups but is destructive for nodes near the common border.

Another argument against the borders visualized in Figure 5.9 (a) is exemplified in the following. Imagine the groups in top and left of group a as well as group b are configured

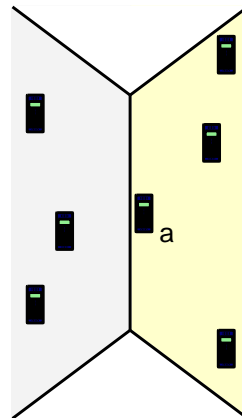


Figure 5.8: Small Changes

with a low transmission power. Since group *a* is surrounded by the other groups, a high transmission power for this group would interfere the communication of all other groups.

The groups *a* and *b* in Figure 5.9 (b) are closer to each other and, therefore, both groups would be configured with more similar settings. Due to the large common border an optimal configuration has to strike a balance between the nodes in the center of a group and those nodes near the border.

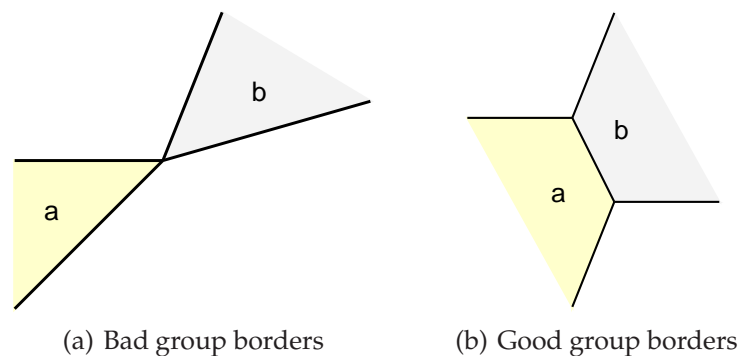


Figure 5.9: Optimal group borders

Summing up the group borders discussion, it can be stated that neighboring groups should not have too different settings. Such a configuration of neighboring groups is a tradeoff between the optimal configuration for nodes in the center and nodes near the border of a group.

To limit the problem concerning nodes at the border of a group, the borders should be as small as possible. In addition to the length of the borders also the amount should be minimal. Based on these two requirements a third criteria can be derived. Groups should not be teathed because such borders are large and nodes near the border are acting as interfering transmitters.

Chapter 6

Group Models

Grouping nodes for adaptation requires the identification of nodes to be configured equally. Since the configuration of nodes is related to several node properties, discussed in Chapter 4, models have to be developed to find groups of nodes with equal properties. The properties, that can be used to identify groups, depend on the algorithm domains to be adapted.

Due to the huge amount of different algorithm domains, this thesis focuses on the development of models to be used for adaptation of routing algorithms.

6.1 Classification

Developing models for systems requires designing algorithms implementing those models. In case of distributed systems, the designers have to think about the location, where the algorithm is executed. In general, there are three possibilities. Running the code on a central component, distributed on several nodes or using a hybrid approach.

Another design decision is the question, what metrics to use. Since the basic task of the group formation component is to implement Function 6.1, which assigns every node to a group, a metrics has to be used, which allows to differentiate nodes.

$$f(\text{node}) \rightarrow \text{group_id} \quad (6.1)$$

The last fundamental design decision is about group layout. In general, groups can be connected or distributed. Connected groups means, that two nodes, belonging to the same group, can communicate without using nodes, belonging to other groups, as relays. Distributed groups allow to distribute the group members all over the network.

Figure 6.1 shows the design space with its three dimensions:

- Coordination: Section 6.1.1
- Used Metrics: Section 6.1.2
- Group Layout: Section 6.1.3

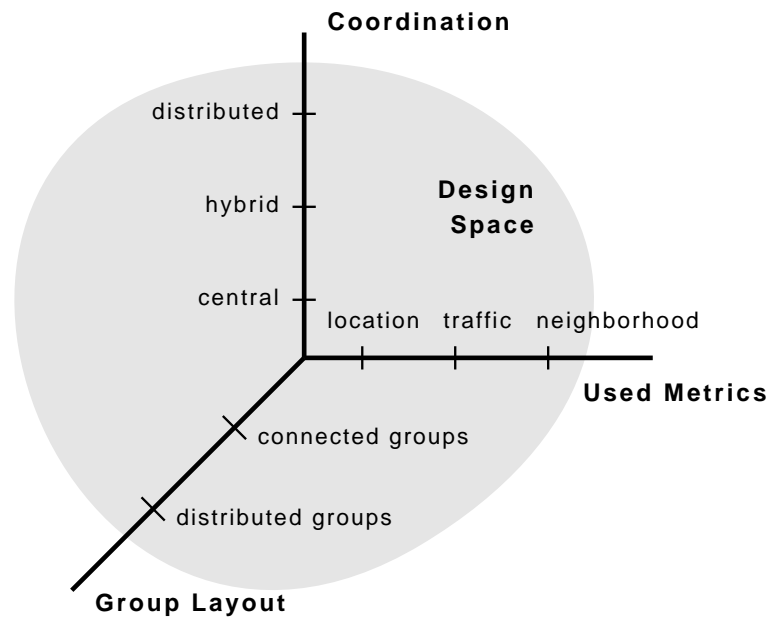


Figure 6.1: Group Formation Algorithm Classification

6.1.1 Coordination

Writing an algorithm running on a central component eases its implementation, because the control flow is not distributed on many nodes. A central approach also enables high optimization possibilities, because the global situation can be taken into account. Having the possibility to use the global situation introduces also the main drawback of the central approach, because the global state has to be transferred to the central component and the nodes have to be informed about their group membership, which implies high effort in large scenarios.

The distributed formation of groups removes the problem of transferring the state to the coordinator and distributing the formed groups back to the network, which allows scalable group formation, even in large scenarios. Due to the fact that all decisions have to be made using local knowledge, no global optimizations can be made, and the created groups are not guaranteed to be optimal.

Combining the benefits of both approaches, global state analysis and distributed group formation, results in the hybrid approach. The idea behind this approach is to analyze the global situation by a central component which calculates parameters for the distributed algorithm forming the groups. Since the central component only calculates parameters, this component does not require knowledge of every nodes' state and, therefore, the amount of data to transfer the global state can be reduced by using aggregation.

6.1.2 Used Metrics

The used metrics is heavily dependent on the domain of the algorithms to be adapted, because these metrics are used to identify those nodes that should be configured equally. In case of routing algorithms the traffic, neighborhood and location of nodes are suitable. A general discussion about these properties is provided in Chapter 4.

6.1.3 Group Layout

When using the formed groups to adapt algorithms, an adjustment between neighboring groups may be required. In case of such a required adjustment, every group has to elect a cluster head for efficient communication between groups, because if every node of one group communicates with every node of the other group, the communication effort would be much too high.

An efficient election of a cluster head and the communication between the cluster head and its group members requires connected groups. Distributed groups do not allow efficient communication between the cluster head and its group members.

6.2 Distributed Approach

As outlined in Section 6.1 there are distributed and centralized solutions to form groups. In the following sections the distributed approaches are discussed in detail. Since there is no central coordinator, the implementation of the models is executed on every node.

There are two possible attempts implementing a distributed group formation model:

- Select one neighbor and create groups by merging the own group with the group of the neighbor: Section 6.2.1
- Join one of several predefined groups based on properties of the node: Section 6.2.2

6.2.1 Most Equal Neighbor

The first distributed model for group formation is based on the idea, that two neighboring nodes with nearly equal properties should be configured in the same way. Since all members of a group are configured equally, these two nodes should be in the same group. Following that basic idea, the same argument can be applied to groups. If two neighboring groups have the same properties the two groups should be merged.

The formation of the groups is split in two phases. In the first phase every node calculates its *Node Property Value*, which maps the properties of a node to a single integer value. This value is then broadcasted to the local neighborhood. After that, every node knows its neighborhood and the *Node Property Value* of the nodes in the neighborhood. Based on the own and the *Node Property Values* of the neighborhood a node selects the neighbor with the most equal value. If two neighbors have the same values, the neighbor with the lower ID will be selected. Every node informs its selected neighbor.

In the next phase a cluster head for the created groups has to be elected. The election is performed using the algorithm introduced in Section 5.2.2.

6.2.1.1 Location

After the theoretical discussion of the *Most Equal Neighbor* this section discusses the calculation of the *Node Property Value* based on the location of nodes. Every node is grouped with its most nearby node. Figure 6.2 (a) shows the *RSSI* values (receiver signal strength indication) of all possible communication links. As outlined in Section 5.1.1 higher values mean a more powerful signal and, therefore, the sending node should be close to the receiver. By selecting those links with the lowest values, the groups, visualized in Figure 6.2 (b), are created.

6.2.1.2 Neighborhood

The second model based on the *Most Equal Neighbor* approach uses the *Neighborhood Value* discussed in Section 5.2.3. Every node builds a group with its neighbor having an equal sized neighborhood. Figure 6.3 (a) shows the sample network with the sizes of the nodes' neighborhoods. By using the links between nodes with most equal neighborhood sizes, the groups, visualized in Figure 6.3 (b), are formed.

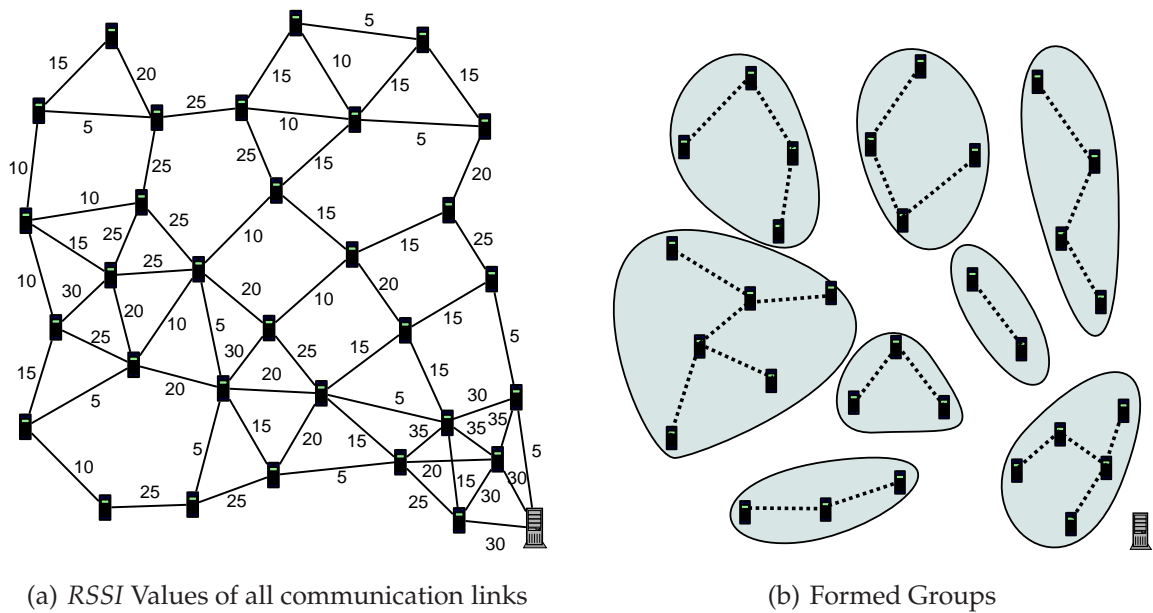


Figure 6.2: Most Equal Neighbor: Location

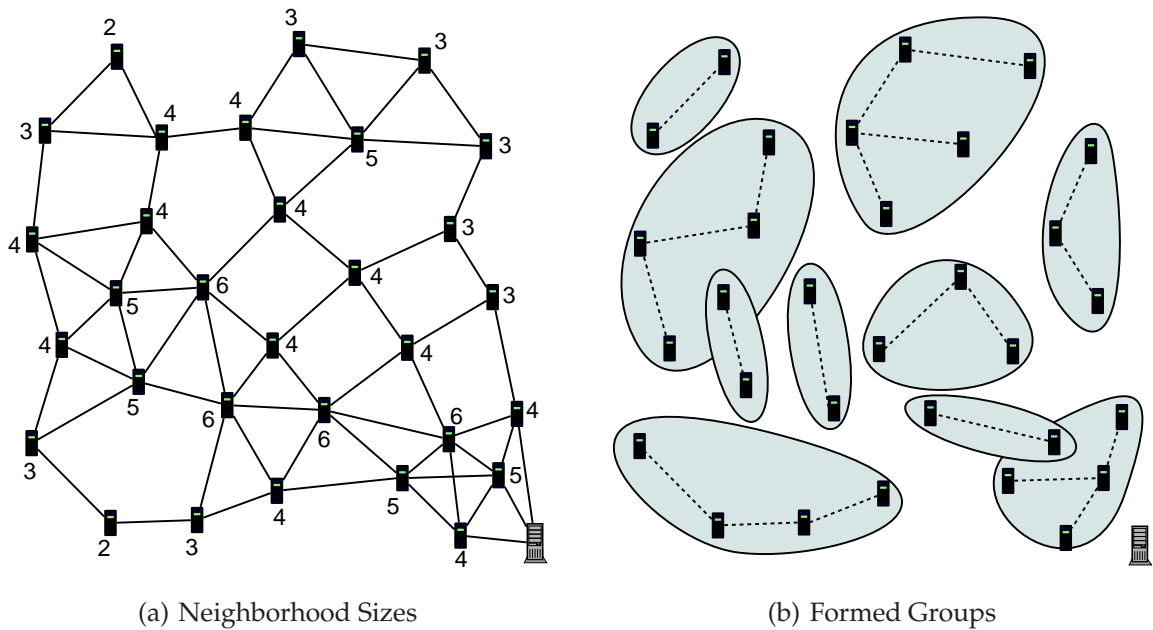


Figure 6.3: Most Equal Neighbor: Neighborhood

6.2.1.3 Traffic

Another possible metrics to select the most equal neighbor is based on the amount of traffic, processed by a node. Figure 6.4 (a) shows a possible routing tree and the traffic at the nodes. When using common routing protocols, the routing tree is not stable, because small changes in the environment can influence the signal strengths. These changes result in changing

node neighborhoods and the routing tree will be reconfigured. Due to the fact that the traffic values cover the received and the transmitted traffic and, therefore, also takes the overheard traffic into account, changes in the routing tree change the traffic pattern marginally. By selecting those links with the most equal traffic, the groups, visualized in Figure 6.4 (b), are created.

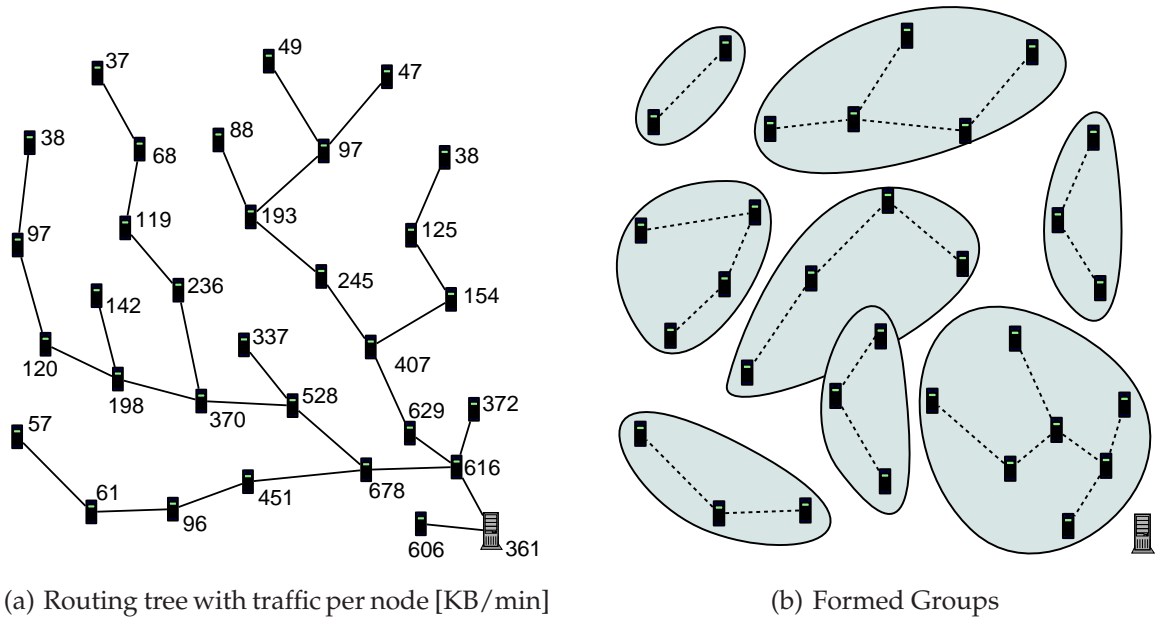


Figure 6.4: Most Equal Neighbor: Traffic

6.2.1.4 Variations

The introduced algorithm only focuses on the most equal neighbor. In scenarios where pairs of nodes have equal properties, such a behavior can result in very small groups. Another problem is the situation where two pairs of nodes have similar properties. Since the nodes belonging to the same pair are more similar than nodes of different pairs, two groups are formed. Due to the similarity between the properties of the nodes, all nodes should build one group.

A possible solution to avoid the fragmentation of groups is to take not only the most equal, but rather the two most equal neighbors into account. Following this approach results in the problem that nodes are building groups with other nodes, even if their properties are different. A more sophisticated solution uses the second equal neighbor only if the similarity to this node is on the same level as to the most equal neighbor.

$$use_second_neighbor(threshold) = \begin{cases} true & \text{if } abs\left(\frac{P_{node} - P_{most_equal}}{P_{node} - P_{second_equal}}\right) = 1 \pm threshold \\ false & \text{else} \end{cases} \quad (6.2)$$

Formula 6.2 shows a possible implementation of this variation. The second equal neighbor is only used if the quotient of the differences between the nodes and its neighbors is close to 1.0. In other words the properties of the second equal neighbor must not be too different. The maximum difference is configured by the variable *threshold*.

6.2.2 Distributed Groups - Static Table

Another distribute approach forms, in contrast to the other approaches, distributed groups. The sensor nodes belonging to such a group are distributed over the whole network and, therefore, this model allows to group all nodes, that have the same properties, which are relevant for the algorithm to be adapted, independently from the location of nodes. In case of routing algorithms the neighborhood of nodes is very important, when adapting the transmission range of nodes.

The underlying idea of this group formation approach is to map the relevant node information to a single integer value. Additionally to this Mapping Function 6.3, a second Mapping Function 6.4 exists, which maps the result of $f_{property}$ to a group identifier.

$$f_{property}(node) \rightarrow integer \quad (6.3)$$

$$f_{group}(integer) \rightarrow group_id \quad (6.4)$$

The first benefit of this group formation model is that it does not require any additional messages to be transmitted and, therefore, the required energy is minimal. The model also allows to group nodes from any part of the network, which results in a high amount of transmitted messages when using other models. Another benefit of this model is the fact, that it can handle small and large groups.

The simple structure of this model, there are only two mapping functions, is its drawback, too. Since there is no communication between the nodes, the target domain of Mapping Function 6.3 has to be a fixed interval, e.g. the interval between zero and one hundred. Without a fixed interval the second Mapping Function 6.4 cannot map sub intervals to groups. With a fixed interval ($min_i..max_i$), it is possible to define $f_{group}(x) = x * number_of_groups / (max_i - min_i)$, where nodes are mapped to $number_of_groups$ groups. Like every introduced distributed approach, this model does not guarantee that each group has the same size, nor that each group has members.

6.2.2.1 Neighborhood Value

When applying the *Distributed Groups* model to the routing algorithms domain, the neighborhood of a node is an adequate candidate for usage in $f_{property}$. As discussed in Section 4.1.3, a smaller neighborhood causes, when sending data, smaller energy consumption. In contrast to this, a larger neighborhood allows more degrees of freedom when creating a routing tree.

Figure 6.5 (a) shows a deployed sensor network with *Neighborhood Values*. As discussed in Section 5.2.3, the *Neighborhood Value* is calculated by counting the number of neighbors. The Mapping Function 6.4 defines five groups, by splitting the interval 0..10 into five equal sized subintervals. The maximum value of 10 and the number of formed groups are constants

defined by the model. Since no central coordinator exists, it is impossible to dynamically calculate these values. Based on these constants, the groups, visualized in Figure 6.5 (b), are formed.

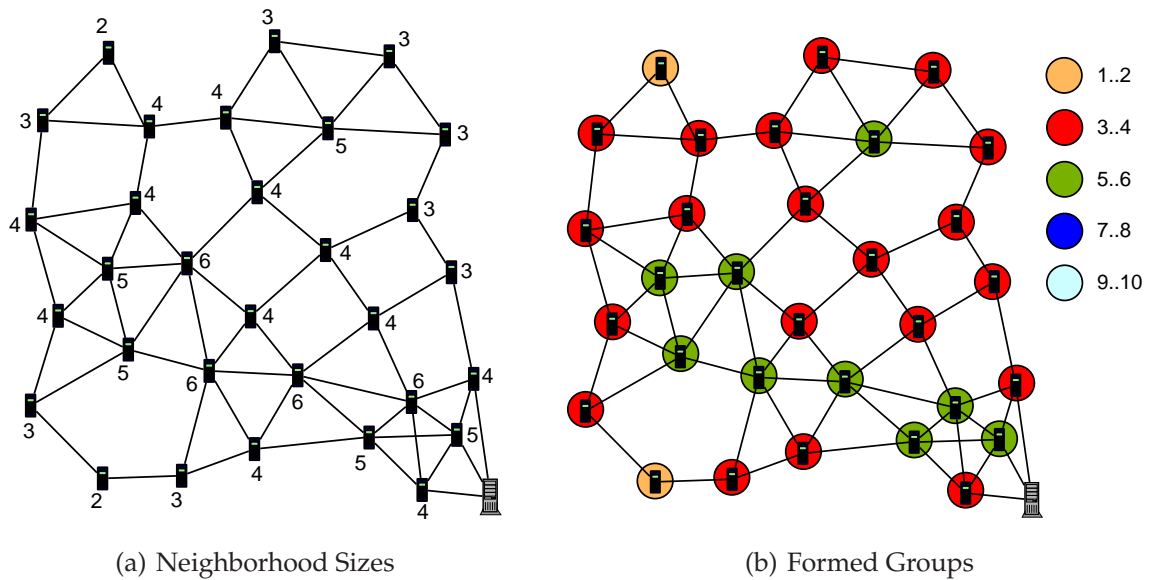


Figure 6.5: Distributed Groups Static Table - Neighborhood

6.2.3 Mobile Groups

The models, discussed in the previous sections, are based on the assumption that nodes are static or moving very slowly. In scenarios where nodes are moving faster, the introduced models fail, because they assume static node properties. When applied to mobile nodes, these models would require a continuously reconfiguration of the groups, which results in high effort and, therefore, in a high energy consumption.

In scenarios where nodes are moving in groups, as introduced in Section 4.2.2, nodes in the same moving group should be grouped for adapting their components. The models discussed in this section detect moving groups by observing the connection times between nodes.

Figure 6.6 (a) shows a two lane motor-way with a drive-up. The nodes are installed on cars moving on the lanes in directions indicated by the arrows. The connection times, visualized in Figure 6.6 (b), are the result of nodes moving in the same direction.

In general, groups are formed by selecting some links between nodes. All connected nodes are forming one group. In the following subsections different approaches to identify those links are introduced.

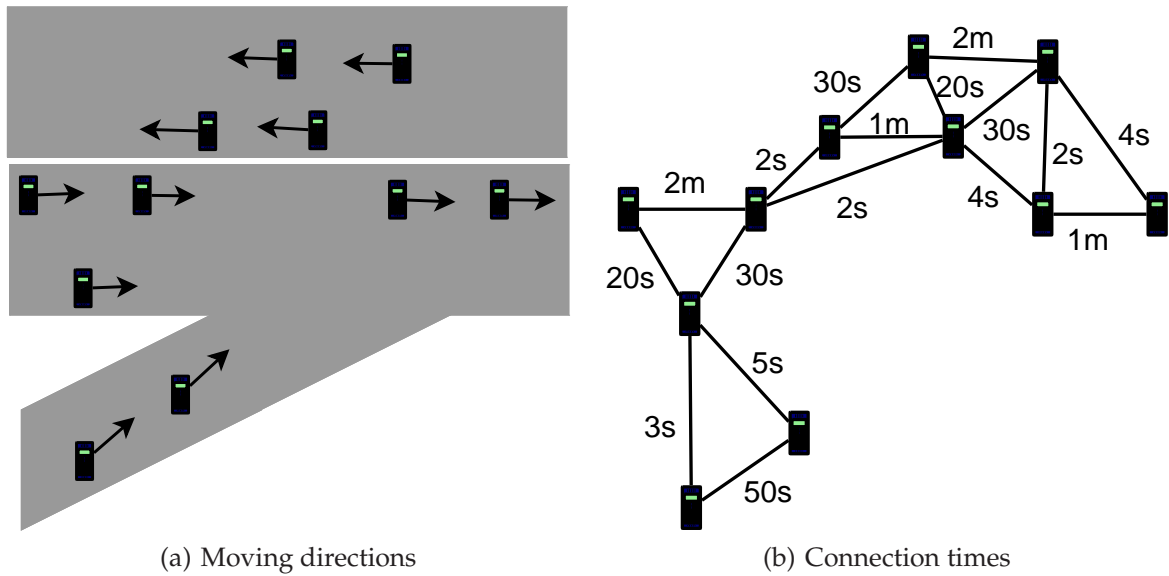


Figure 6.6: Moving Nodes

6.2.3.1 Maximum Connection Time

The first approach to identify links is based on the idea, that every node joins the group of those neighboring nodes, which is in the neighborhood for the longest time. Following this approach every node selects its neighbor with the maximum connection time between them.

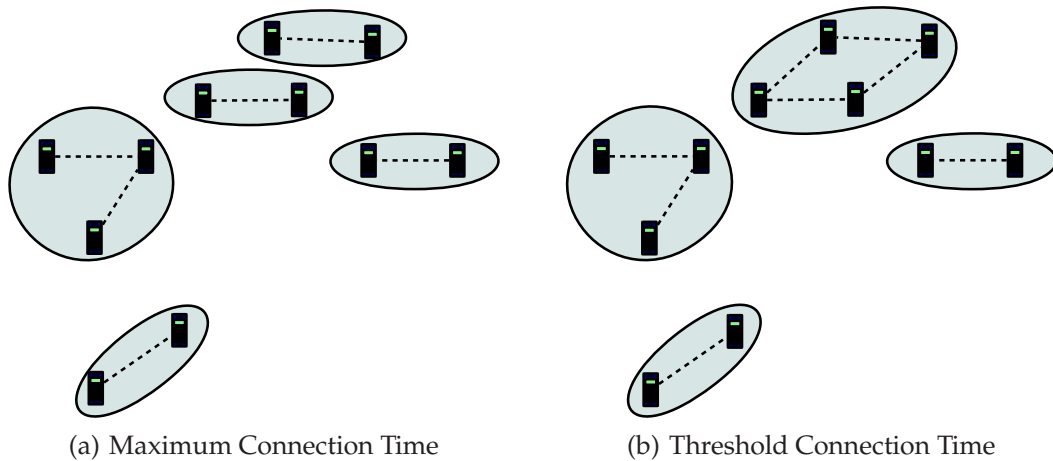


Figure 6.7: Metrics using Connection Time

Figure 6.7 (a) shows the formed groups, created by applying the *Maximum Connection Time* approach to the network visualized in Figure 6.6 (b). When surveying the created groups it stands out that the groups only cover few nodes. Especially the top four nodes are split into two groups, although all nodes are moving in the same direction.

This phenomenon is not restricted to this sample network. Consider a network containing two pairs of nodes, where the nodes belonging to the same pair have known each other a long time. In such a network the two pairs are always grouped to different groups. The problem arises from the fact that the nodes in each pair select the same link and, therefore, each pair creates its own group. Every other node in the network has two choices: Join one of the two groups or create/join another group.

To conquer this drawback the next section introduces an approach, where every node can select more than one link, which allows to merge two groups.

6.2.3.2 Threshold Connection Time

The idea behind the *Threshold Connection Time* approach is the fact that one node should share a group with all its neighbors, where the link between them is active at least for a specific time period. The length of this period depends on application knowledge. One factor influencing the threshold is the maximum node speed, because in case of fast nodes the connection times between nodes are generally smaller than in scenarios with slow nodes. Finding the optimal threshold requires the simulation of the application and group formation component with different thresholds.

When analyzing the groups, see Figure 6.7 (b) formed by the *Threshold Connection Time* approach, it stands out that the top four nodes are grouped together.

Summing up the properties of the two approaches, the *Threshold Connection Time* approach groups all nodes, belonging to the same moving group, while the *Maximum Connection Time* approach splits the moving group to several groups. The benefit in the created groups has to be payed with an additional parameter, that has to be configured on every node.

6.3 Hybrid Approach

The main benefit of distributed approaches to form groups is scalability, because the models, following this approach, only use local knowledge. This approach eliminates sending the state of nodes through the network, where a centralized component calculates the groups.

Taking the model *Distributed Groups - Static Table*, discussed in Section 6.2.2, as an example for a distributed approach, a weakness becomes visible. Using only local knowledge does not guarantee equal sized groups. A fundamental problem is the static table, which has to cover all potential values and not only those values that have been assigned to nodes.

Hybrid approaches try to combine the benefits of distributed and centralized approaches. A central component calculates parameters, based on the global situation and a distributed algorithm forms groups using those parameters.

6.3.1 Distributed Groups - Dynamic Table

When applying the hybrid approach to the *Distributed Groups - Static Table* models, the static table is replaced by a dynamic table. The dynamic table is calculated by a central component. In contrast to the centralized approaches, here it is not mandatory here to transfer the total state of each node to that component. In the most trivial case only the minimum and maximum value is transferred, which enables the possibility of aggregation, to limit the required traffic. Based on these two values, an optimized table can be created. Since the table is adapted to the values which are actually present in the network, the group sizes are more similar.

After creating the table, all nodes have to be informed about the calculated table. Flooding is used for distribution and, therefore, every node has to transmit the table once.

6.3.1.1 Neighborhood Value

A possible property for group formation, is the Neighborhood Value, discussed in Section 5.2.3. Figure 6.8 (a) shows a sample network with the Neighborhood Value at each node. Using the minimum and maximum value, the base station can build the table consuming the intervals belonging to each group. The table and the created groups are visualized in Figure 6.8 (b).

6.3.1.2 Traffic

In Figure 6.9 (a) the traffic at each node is visualized. By sending and aggregating the minimum and maximum traffic, the table, used for group formation, can be created. After distributing the table, every node can join its group. The formed groups are illustrated in Figure 6.9 (b).

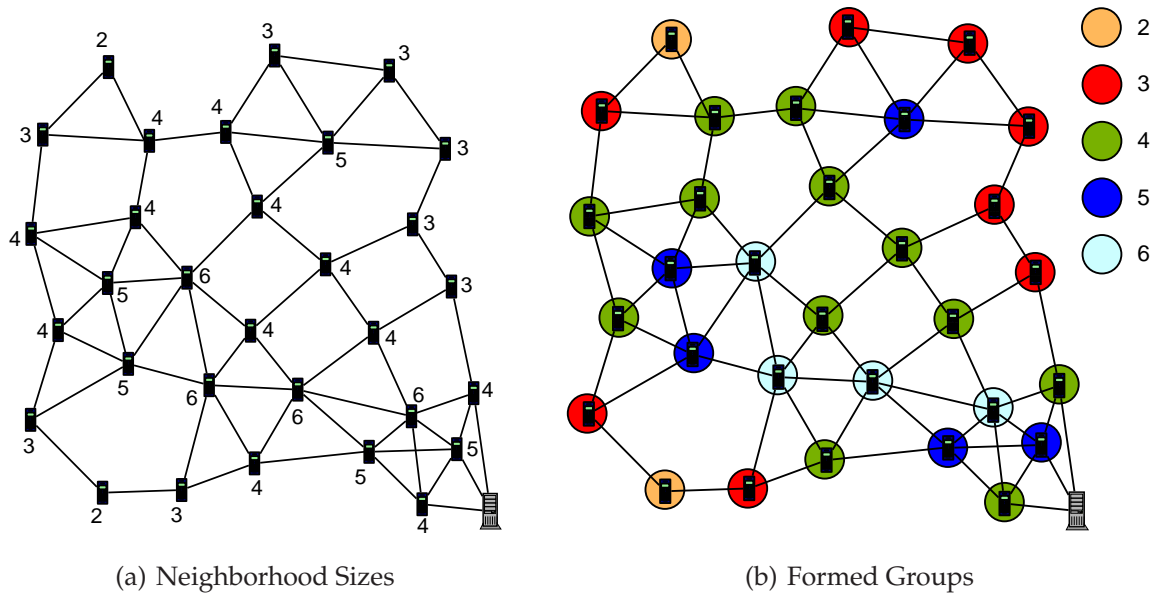


Figure 6.8: Distributed Groups Dynamic Table - Neighborhood

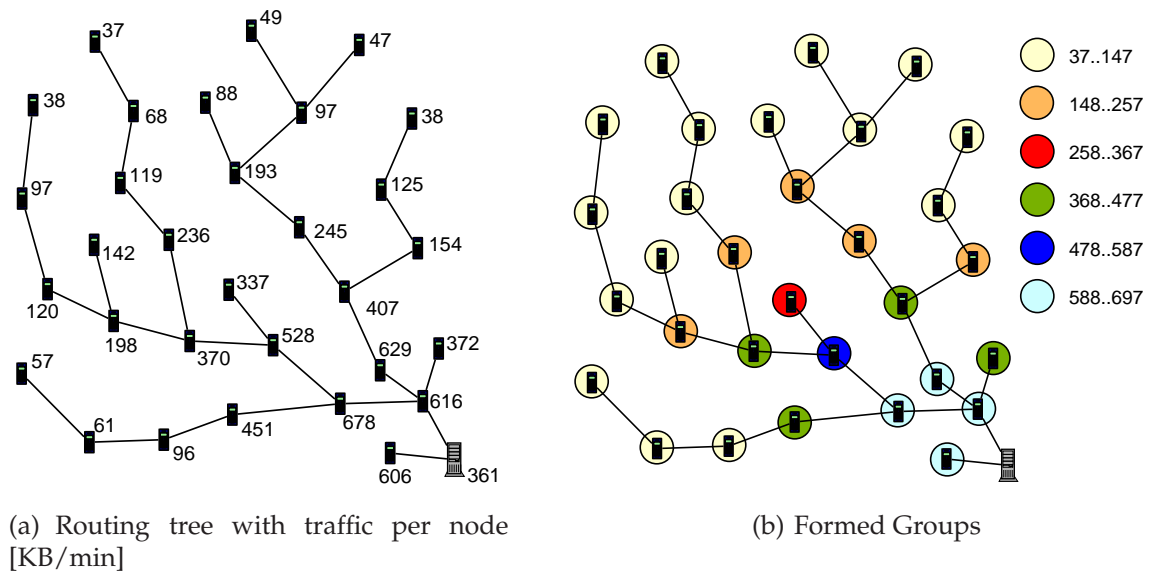


Figure 6.9: Distributed Groups Dynamic Table - Traffic

6.3.2 Location Based

In sensor networks the workload at nodes is closely related to the location in the network. Taking the expected traffic as a workload example, this circumstance becomes obvious. Section 4.3.1 explains the relation of traffic and node location in detail. Based on location information of nodes, it is possible to form groups with equal behavior and, therefore, these groups can be used for adaptation.

The group formation process, based on location information, is divided in several steps. Since some of these steps are performed by a central coordinator and others can be performed by nodes themselves, this approach is a hybrid approach. The different steps are:

1. **Collect information about node distribution:** In the most trivial case the minimal rectangle containing all nodes is calculated. In general, different densities in the network can be sent to the base station, too. A coordinator knowing the position of every node would be the best result of this step. When using *Network Status Monitoring Frameworks*, see Section 5.1.2, this task can be performed without sending any additional messages.
2. **Define a pattern for virtual cluster heads:** The second task is executed by the central coordinator. Using the node distribution information, the coordinator defines a set of virtual cluster heads. Every virtual cluster head corresponds to a group of nodes because the neighborhood of a virtual cluster head defines the group. Generally the virtual cluster head can be freely placed, but taking the distribution of the virtual cluster heads' position into account, a regular pattern results in less distribution effort.
3. **Distribute the pattern in the network:** After the pattern definition, all nodes in the network have to be informed about the used virtual cluster head positions. The distribution can be performed by flooding the pattern through the network.
4. **Determine the closest virtual cluster head:** Every node knows the position of every virtual cluster head, which allows the calculation of the nearest virtual cluster head. Nodes selecting the same cluster head belong to the same group.
5. **Run cluster head election:** Every group of nodes now determines a "real" cluster head. To elect the cluster head the algorithm, introduced in Section 5.2.2, can be used, with little adaptations. To limit the election to nodes, belonging to the same group, every sent message contains a virtual cluster head identifier. Only messages, where the identifier of the message and the own virtual cluster head are equal, are used. The second adaptation is the fact that the introduced algorithm requires all nodes to be arranged in a tree. This limitation can be removed by building a tree during the *Explosion Phase*.

As outlined in the general discussion of the location based approach, the pattern used to define groups is not optimal. Exact knowledge of node positions or too much effort on the pattern distribution forbids that. Due to the suboptimal location of virtual cluster heads it cannot be assumed that all nodes, having the same nearest virtual cluster head, are connected. In such case the election algorithm will create two cluster heads and, therefore, two groups are formed. Since the two groups cannot communicate with each other, the failure of the pattern results in the benefit that both groups can adapt its setting individually.

Figure 6.10 shows a sample network. The location of the virtual cluster heads are marked by circles. Every node selects the nearest virtual cluster head, visualized by the dashed lines. Grouping nodes with the same virtual cluster head results in the drawn groups.

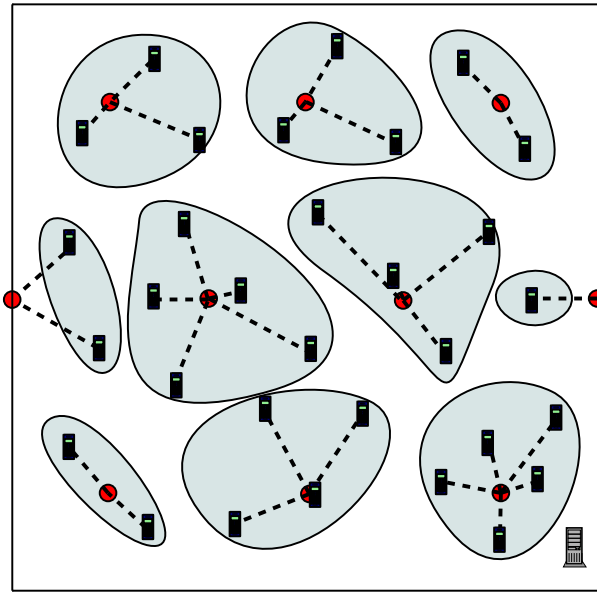


Figure 6.10: Generic Location Based Model

6.3.2.1 Hexagon Pattern

The pattern used to form groups is dependent on the algorithms to be adapted. The behavior of some algorithms is related for example to the density of nodes. Since the location based approach is a general approach and, therefore, not limited to a specific domain of algorithms, this section introduces a pattern for general purpose.

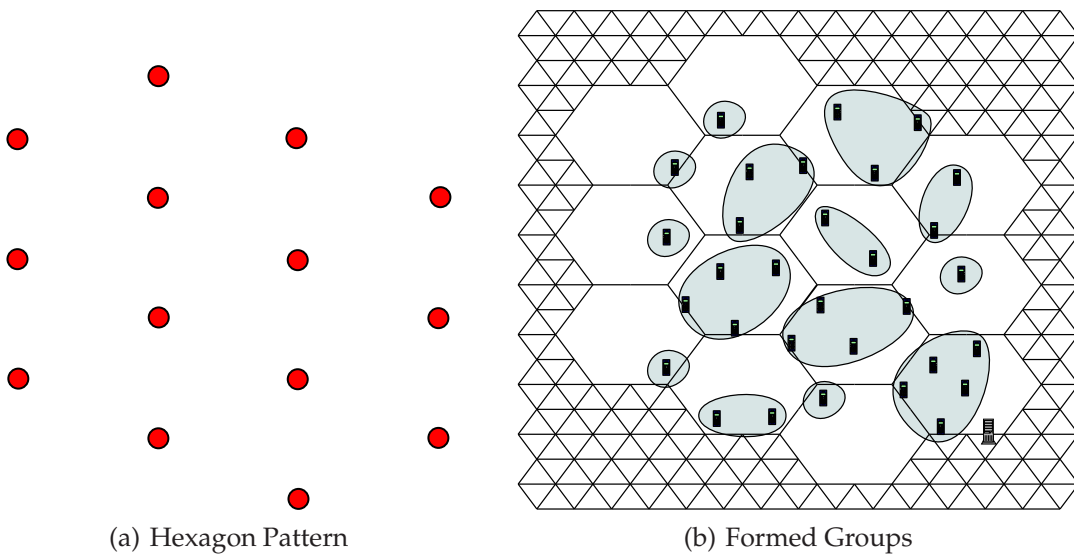


Figure 6.11: Location Based Model: *Hexagon Pattern*

The optimal form of a group is the circle, because this form allows minimal communication effort between the cluster head and the group members. The purpose of the groups is to assign different groups different settings. Different settings between nodes often complicate the communication between those nodes, and the effect is amplified by the amount of differences between the settings. This statement becomes obvious when taking the transmission power as an example. Based on this statement, nodes of different groups should only be connected, if both groups have similar settings.

The *Hexagon Pattern*, visualized in Figure 6.11 (a), results in groups with a hexagon shape, which is similar to the required circle shape. The second benefit of this pattern is that the groups are compact and, therefore, the differences between groups are limited. When applying the *Hexagon Pattern* to a sample network, the groups, showed in Figure 6.11 (b), are created.

6.3.2.2 Circular Pattern

Since the group formation models discussed in this thesis focuses on the routing domain, the second introduced pattern uses domain knowledge. In scenarios where data is routed to the base station, the traffic processed by a node increases with the closeness to the base station. Based on the assumption that the behavior of routing algorithms at a specific node is related to the amount of traffic at that node, it is adequate to configure nodes depending on the distance to the base station.

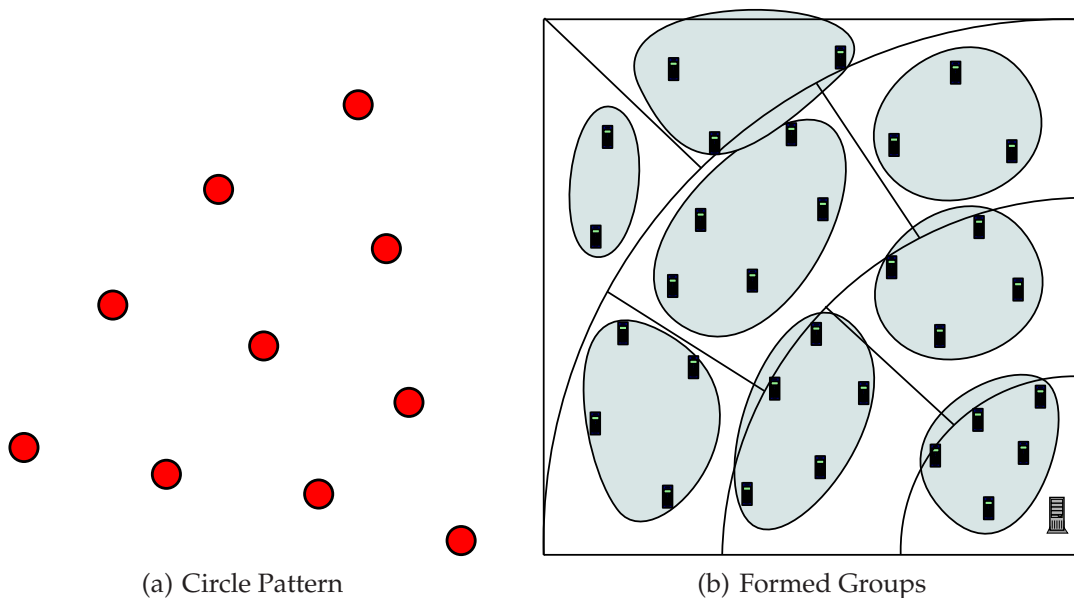


Figure 6.12: Location Based Model: *Circular Pattern*

Using only the distance property to form groups results in small groups with low base station distance and large groups with high base station distance. The *Circular Pattern* arranges the virtual cluster heads, see Figure 6.12 (a), on circles around the base station, which results in equal sized groups. Figure 6.12 (b) shows the groups defined by the *Circular Pattern*.

6.4 Centralized Approach

The third category of group formation approaches are centralized models. In contrast to the distributed and hybrid approaches, the groups are formed by a single algorithm running on the base station. Based on this idea, the group formation is divided into three phases:

1. Collect node information and transfer them to the base station
2. Run group formation algorithm and assign cluster heads
3. Inform cluster heads about their members

6.4.1 Weighted Links - Graph Partitioning

The approach, discussed in this section, is based on the idea of weighting the links between nodes. The calculation of the weights depends on the properties of the involved nodes. Using the graph partitioning algorithm, discussed in Section 5.2.1, the graph is divided in several subgraphs, which contain the nodes to be grouped.

The following derivatives of the *Weighted Links - Partitioning Graph* approach are using different node properties to weight the links. The first approach is using the neighborhood of the nodes, while the second approach is based on the traffic at the nodes.

6.4.1.1 Neighborhood

Figure 6.13 (a) shows a deployed sensor network with edges E between nodes N where communication between the nodes is possible. Every edge e_{ij} (edge between node i and j) is initially assigned the *RSSI* value (received signal strength indication, normalized to the interval 1..100, where 100 stands for optimal communication).

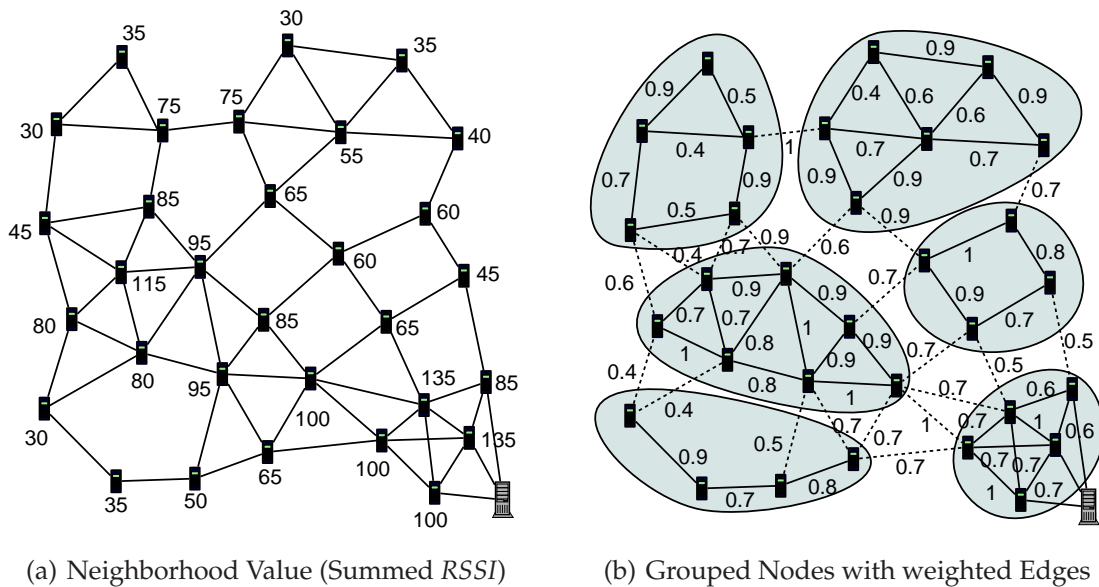


Figure 6.13: Weighted Links - Graph Partitioning: Neighborhood

In the next step the *Neighborhood Value* (v_k , *Neighborhood Value* of node k) is calculated by Formula 6.6, where the *RSSI* values of the node's communication links are summed up. This calculation schema results in high *Neighborhood Values* for nodes with a large neighborhood. Since usage of long distance links requires large transmission power and, because of this, consumes a lot of energy, the usage of these links should be avoided, which enables the possibility to configure the nodes with a lower power value.

Since the problem of the group models is to create groups, partitions of nodes have to be created and, therefore, some edges have to be removed. One of the optimal group properties is to form groups of nodes with as equal as possible properties and, because of this, those edges between nodes with different neighborhood values are removed. The weight of the edges e'_{ij} is recalculated (Formula 6.7) by assigning the minimum quotient of the *Neighborhood Values* of node i and j . The calculation results in values between one and zero ($e_{ij} = 1$ if $v_i = v_j$).

$$e_{ij} = RSSI_{normalized}(i, j) \quad (6.5)$$

$$v_k = \sum_{e_{ij} \in E \& i=k} e_{ij} \quad (6.6)$$

$$e'_{ij} = \min(v_i, v_j) / \max(n_i, n_j) \quad (6.7)$$

$$C(G) = \sum_{e_{ij} \in E \& n_i \in G_{l_1} \& n_j \in G_{l_2} \& l_1 \neq l_2} e'_{ij} \quad (6.8)$$

The groups, required for adaptation, are formed by partitioning the graph $G(N, E')$ in t groups g_l (with $\frac{|N|}{t} \pm \varepsilon$ nodes per group), while minimizing the cost function listed in Formula 6.8. This function covers those edges, where the connected nodes are assigned to different groups. Figure 6.13 (b) visualizes the formed groups. The edges drawn with a dashed line, are identified by the graph partitioning algorithm for removal.

6.4.1.2 Traffic

In the sensor network, showed in Figure 6.14 (a), the traffic at the nodes is visualized. Following the steps, discussed in the previous section, the link weights are calculated, based on the traffic. Applying the graph partitioning algorithm, groups can be formed, see Figure 6.14 (b). The identified edges for removal are drawn as dashed lines.

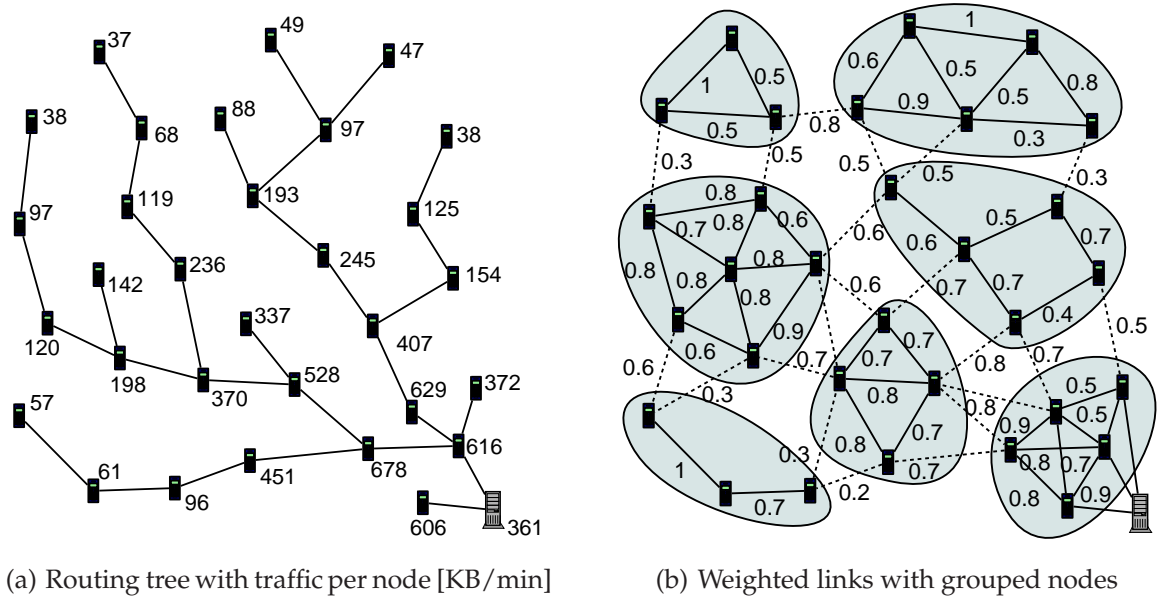


Figure 6.14: Weighted Links - Graph Partitioning: Traffic

6.5 Model Extensions

The discussion of the different group formation models was based on a sample network with equal nodes and one base station. As discussed in Chapter 4 a sensor network in real applications can differ from that sample. The intention of this section is to introduce extensions for the discussed models to support more complex networks, like multi base station scenarios.

In contrast to extensions, which extend the target domain of the models, other extensions can be used to optimize the behavior of the introduced models. These optimizations can be applied to several models and, therefore, they are discussed together.

6.5.1 Multi Base Station Model

The sample network used for the discussion of the models has only one base station. In several deployed networks this assumption cannot be made and, because of this, this section discusses required modifications on the introduced models.

When analyzing the different models, it becomes obvious that most of them support multiple base stations out of the box. All distributed models are using the traffic or the neighborhood of nodes to determine group memberships. When using the traffic metrics multiple base stations are taken into account, because the traffic at nodes results from the base station location. In case of the neighborhood metrics, no adaptations are required, because the density of nodes is independent from the number of base stations and, therefore, no assumptions about the number are made.

In the category of hybrid approaches some adaptations have to be made. The first approach

of that category is using a dynamic calculated table. Since the calculation of that table is initialized by collecting the status of sensor nodes by a central component, there are two possible modifications, depending on the existence of an infrastructure between the base stations.

In case of a present infrastructure between the base stations, the collection can be performed by using the existing trees. Each base station collects the status of its network part and shares the information with the other stations using the infrastructure. The calculation of the table can be performed on each station or the stations have to elect one master station. The created table can be distributed to the network using flooding.

In case of no infrastructure is present, a master base station has to be configured on the sensor nodes. The algorithm to create the table can be performed as discussed in Section 6.3.1 by using only the master base station.

The second hybrid approach uses location information to form groups. The modifications on this model are analog to the previous discussed models. Depending on the existence of an infrastructure, the collection of the node location information and the calculation of the pattern describing the location of the virtual cluster heads, can be performed on a single master station or using the infrastructure.

When forming the position of the virtual cluster heads the different base stations have to be taken into account. In case of the circular pattern, the cluster heads have to be arranged on circles around multiple base stations.

Central approaches also have to be modified analogically to the hybrid approaches, because the status of the nodes has to be collected, too. Since the traffic and neighborhood metrics are used to calculate the link weights, no additional modifications are necessary.

6.5.2 Environment Based Model

The communication between nodes is heavily influenced by the environment. Walls or other obstacles limit or forbid communication between nodes. In other scenarios communication is limited to specific regions in the environment caused by application logic. As discussed in Section 4.2.3, the environment should be taken into account when forming groups of nodes.

Figure 6.15 shows a sensor network, deployed in a house. Assuming the walls shield all electromagnetic waves, communication is only possible between nodes in the same room, or through doors between rooms. The introduced limitations enforced by the environment, suggest to group nodes in the same room.

The focus of this section is to analyze if the introduced models are suitable to consider the limitations enforced by the environment. All approaches based on the neighborhood or weighting the links between nodes are supporting the introduced scenario, because there exist only links between nodes in the same room or through doors. Assuming that the walls only attenuate the signal and, therefore, links between nodes of different rooms exists, traffic based approaches require little adaptation. Links between nodes should only be considered, when the quality (*RSSI*) of the link is above a given threshold.

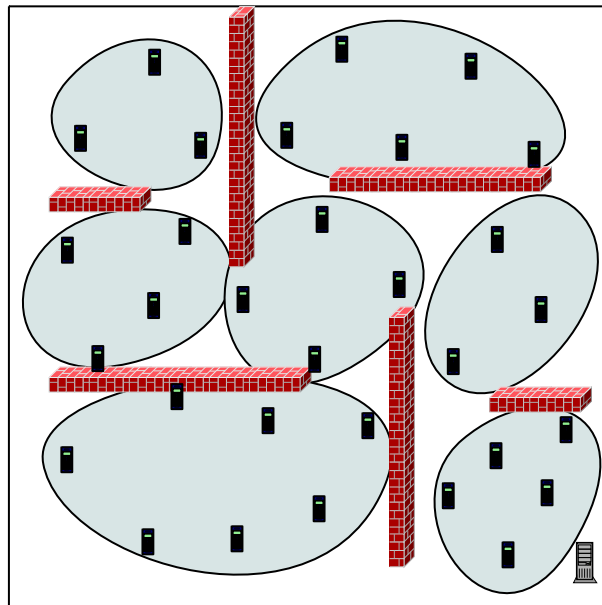


Figure 6.15: Environment Based Model

The distributed approaches are generally suitable for the introduced scenario, because they do not create connected groups. The remaining approach is based on location information. By definition this approach is optimal suitable to support such scenarios, because the virtual cluster heads can be distributed to the different rooms, which forms the groups visualized in Figure 6.15.

Summing up the environment driven groups, it can be stated that all introduced models support scenarios where communication is limited by obstacles.

6.5.3 Different Node Types Model

When comparing the sample network used for the model discussion with networks used in a real application, like the observation of bridges [MSKG05] a fundamental difference becomes visible. The sample network consists of several homogeneous nodes while in real applications networks with different nodes are used, too. This section discusses possible problems when applying the introduced models to heterogeneous networks.

Taking the observation of bridges as an example, this application requires information about temperature of the bridge and the occurrence of vibrations within the building. Collecting these two properties requires fundamental different sensors. The temperature can be easily measured by a node with an attached temperature sensor. Detecting and locating a vibration source requires several nodes with motion sensors. By exchanging the collected motion signal, the nodes can determine the location, where the vibration was created.

Since the temperature does not change very fast, the interval between the measurements can be long. To save energy the nodes can sleep during the intervals. Since possible vibra-

tions appear randomly, nodes with a motion sensor cannot sleep in predefined intervals. Keeping the different sleeping times in mind, it becomes obvious, that these two types of nodes should not be mixed in groups, used for adaptation.

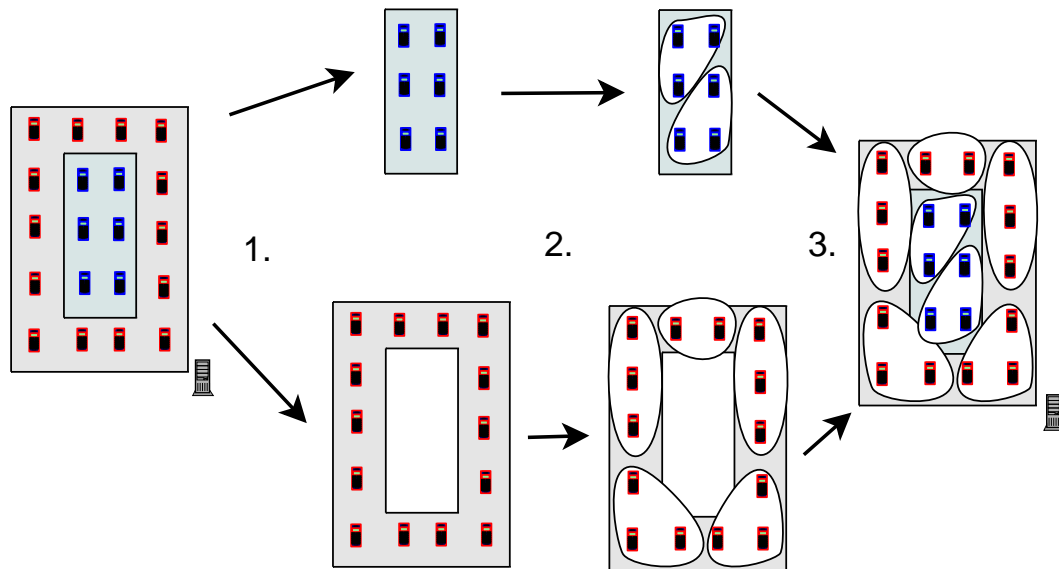


Figure 6.16: Different Node Types Model: 1. split network in subnets with nodes of same type; 2. apply group formation in each subnet; 3. merge subnets

Figure 6.16 shows how to extend the introduced models to support networks with different node types. The visualized sample network contains six sensors of type *blue* (in the center) and 14 sensors of type *red* (at the edges). During the first phase the network is partitioned in two networks containing only nodes of same type. The introduced models for group formation can now be applied to each network partition. The result of the second phase are grouped nodes for each network partition. Using the created groups of both network partitions and applying them to the original network, the required groups are created.

6.6 Theoretical Model Discussion

In the previous sections multiple group formation models have been introduced and discussed in detail. Since the models were discussed on their own, independent from other models, this section gives an overview on all models. In general, the models are compared with each other. The comparison covers the fulfillment of the optimal group properties, introduced in Section 5.3, as well as the fulfillment of other requirements based on the sensor network properties, see Chapter 4. The used criteria are as follows:

- **Density Consideration:** For a detailed discussion see Section 5.3.1.
- **Traffic Consideration:** Section 5.3.2 contains a detailed discussion.
- **Group Borders:** See Section 5.3.3 for a detailed discussion.
- **Configurable Group Sizes:** Covers the question, whether it is possible to configure the size of the created groups, or if the size is a result of the actual node layout.
- **Mobility Support:** Covers the question, whether the model can be used to form groups in mobile scenarios.
- **Connected Groups:** Whether the formed groups are connected, is covered by this criteria.
- **Required Messages:** Group formation requires messages to be transmitted in the network. This property determines the size of the introduced overhead. The *Required Messages* are calculated assuming that the nodes are randomly deployed in a rectangle and, therefore, the number of messages represents the average case.

During the discussion of sensor network properties, see Chapter 4, additional features have been identified. These features are for example:

- Multiple Base Stations
- Environment Consideration
- Different Node Types

As discussed in Section 6.5, all introduced models can be extended to support these requirements and, therefore, they are not covered during the comparison in this section.

Table 6.1 lists the different introduced models and the required features. The fulfilled features are marked using \checkmark , while \circ marks unsupported features. Partially supported features are denoted with the symbol \checkmark/\circ .

Models considering the density of nodes are on the one hand those which are using the neighborhood metrics and on the other hand the *Location Based* models. The fact that *Location Based* models are considering the density, follows from the possibility to place virtual cluster heads depending on the density distribution in the network. When looking at models considering the traffic, the same situation as for the density appears. Models based on the traffic metrics consider traffic directly, and the location based models are considering the traffic indirectly. The indirection results from the fact that the traffic at a node is an effect of the node's position.

Group Model	Density Consideration	Traffic Consideration	Group Borders	Configurable Group Sizes	Mobility Support	Connected Groups
Most Equal Neighbor - Location - Neighborhood - Traffic	○ ✓ ○	○ ○ ✓	○ ○ ○	○ ○ ○	√/○ ○ ○	✓ ✓ ✓
Distributed Groups - Static Table - Neighborhood	✓	○	○	○	○	○
Mobile Groups	○	○	○	○	✓	✓
Distributed Groups - Dynamic Table - Neighborhood - Traffic	✓ ○	○ ✓	○ ○	√/○ √/○	○ ○	○ ○
Location Based - Hexagon Pattern - Circular Pattern	√/○ √/○	√/○ √/○	✓ ✓	✓ ✓	○ ○	✓ ✓
Weighted Links - Partitioning Graph - Neighborhood - Traffic	✓ ○	○ ✓	✓ ✓	✓ ✓	○ ○	✓ ✓

Table 6.1: Group Model Discussion - Supported Features

The feature concerning optimal group borders is supported by the location based models, because the introduced pattern forms more or less circular groups which have borders of minimal length. An argument against these models is the fact that small position changes of nodes near the border result in joining another group. The *Weighted Links* model supports optimal group borders, because the used optimization function minimizes the links between nodes of different groups.

The configuration of fixed group sizes is only possible with the knowledge of the global state. The models using dynamic tables are generally suitable to allow such a configuration. In praxis the fixed group sizes require the creation of a total order of all nodes. The creation of such an order restricts the possibility of using aggregation to limit the amount of data to be transferred to the base station. The *Location Based* and *Weighted Links* models are supporting the configuration of group sizes. When using the *Location Based* approach, the position of every node has to be known by the base station to calculate the positions of the virtual cluster heads, resulting in groups with a fixed size. In case of the *Weighted Links* approach the group size can be configured by a parameter of the algorithm.

Mobility is partially supported by the *Most Equal Neighbor* model using the location of nodes as a metrics, because the model groups nodes with minimal distance to each other. Since the relation of the closeness of nodes and moving groups is minimal, only the *Mobile Groups* model fully supports mobility.

The *Most Equal Neighbor*, *Location Based* and *Weighted Links* models as well as the *Mobile Groups* model are forming connected groups. The remaining models are forming, as the name denotes, distributed groups. Forming connected groups allows to determine a cluster head to adjust the configuration of a group during the adaptation process. In case of distributed groups no adjustment between groups is possible.

Whenever introducing a new software module for deployment in sensor networks, the overhead introduced with the new module has to be identified. The overhead can consist of computing, data storing or message transmission components. In case of group formation models, the amount of computations on sensor nodes is negligible. The amount of data to be stored can be ignored, too. In contrast to the previous overhead components, the transferred messages vary a lot between the different models, and the influence of message transmissions to the network lifetime is significant.

Symbol	Explanation
n	number of nodes deployed in the network
$c(n)$	required messages to collect the status of network; the existence of <i>Network Status Monitoring Frameworks</i> (Section 5.1.2) zeros these messages
$t(n)$	required messages to cover the dynamic table
$p(n)$	required messages to cover the pattern describing the location of <i>Virtual Cluster Heads</i>
$\#groups$	number of formed groups
$g(n)$	required messages to cover the IDs of nodes belonging to one group
$e(n)$	required messages to elect a cluster head for each group. Using the election algorithm, introduced in Section 5.2.2, the effort is $3 * n$ messages.

Table 6.2: Group Model Discussion - Message Overhead Key

Group Model	Required Messages	
Most Equal Neighbor	n	$+e(n)$
Distributed Groups - Static Table	0	
Mobile Groups	n	$+e(n)$
Distributed Groups - Dynamic Table	$t(n) * n$	$+c(n)$
Location Based	$n * p(n)$	$+c(n) +e(n)$
Weighted Links - Partitioning Graph	$\sqrt{n} * \#groups * g(n) + n * g(n)$	$+c(n)$

Table 6.3: Group Model Discussion - Message Overhead

Table 6.3 lists the required number of messages for each group formation model. Since the number of messages is independent from the used metrics, see Section 6.1.2, no additional differentiation of the models is required.

The messages required for the *Most Equal Neighbor* and the *Mobile Groups* approach consists of two components. Every node has to inform its most equal neighbor about its election, which requires n messages. The election of a cluster head requires $e(n)$ messages.

Using the *Distributed Groups - Static Table* is best choice, when taking only the required messages into account, because this approach requires no messages to be transmitted. In contrast to the usage of a static table, using dynamic tables requires several messages. Due to the central table calculation the status of the network has to be transmitted to the base station, which requires $c(n)$ messages. After the calculation the table has to be distributed. Using an existing routing tree, every node has to send the table once which requires $t(n) * n$ messages. The *Distributed Groups - Static Table* and *Distributed Groups - Dynamic Table* approach requires no cluster head and, therefore, the cluster head election is not needed.

The *Location Based* approach calculates the location of virtual cluster heads based on the network state. The effort to collect this state is $c(n)$. The distribution of calculated locations requires $n * p(n)$ messages, because the pattern containing the location information has to be sent once by each node. Since the formed groups are connected, a cluster head is required. Its election needs $e(n)$ messages to be transmitted.

All previous discussed approaches are using distributed algorithms to detect the group membership. The *Weighted Links - Partitioning Graph* approach calculates the membership centralized and, therefore, the state of the network has to be collected by sending $c(n)$ messages. The central component calculates the groups and the cluster head of each group. The information of each cluster head about its group requires $\sqrt{n} * \#groups * g(n)$ messages, because a cluster head has an average distance to the base station of \sqrt{n} hops. Depending on the transmission range of the nodes, this value can be reduced by a constant factor. Now every cluster head has to inform its members which requires $n * g(n)$ messages. Since all nodes in a group are connected, the distribution of the group membership packet can be realized by flooding the list of members by the cluster head. Every node refloods the list, if it is on the list and, therefore, every node sends the list once.

6.7 Summary

In this chapter different group models were introduced. These models are based on different node properties like location and traffic as well as network properties like density. The actual algorithms to form the groups are designed with different coordination patterns. These patterns range from distributed algorithms to models using a central coordinator. Combinations of both approaches build the third type, named hybrid models.

When recapitulating the effort to form the groups using the different models, huge differences become visible. Most approaches require an effort of $O(n)$ messages or in other words a constant number per node. As expected, the number of messages required by the center algorithm is much higher. The model with the lowest overhead requires zero messages to be transmitted. The drawback of this approach is unbalance in the group sizes.

Since this chapter only analyzes the models theoretically, the quality of different approaches cannot be rated yet. In the following chapter the models are evaluated using network simulation. Based on the simulation results, the quality of the models can be determined.

Part III

Evaluation and Conclusion

Chapter 7

Evaluation

In the second part of this thesis several group formation approaches have been introduced and theoretically discussed. Since these models are used to group nodes in sensor networks, the consideration of many different effects, which are influencing the performance or the quality of the formed groups, is implied. These effects are for example: concurrency, signal errors, packet collisions.

Due to these effects it is infeasible to evaluate the designed models only in a theoretical manner. To evaluate the performance of the designed approaches, the models are implemented and simulated using a network simulator, see Section 7.1. Using this approach it is possible to measure the quality of the models in different scenarios, see Section 7.2.

The analysis of the measurement results is subject of Section 7.3. In the first part the benefit of using designed models is determined. The quality of the group formation models is analyzed by performing simulation with individually configured nodes and comparing the results.

The second part of the analysis focuses on statements about node configurations, because using the group formation in real applications requires algorithms to determine the optimal settings for each group.

7.1 Implementation

The evaluation of the designed models using network simulators requires an implementation of the models. Since every simulator abstracts in some way from the real world, this section introduces the used simulator and discusses the implementation of the models.

Figure 7.1 gives an overview of the test environment and the used components. The network simulator *CUBUS* [CUB07], developed at the University of Stuttgart, is used as test environment. Section 7.1.1 discusses the usage of this simulator.

The main components used for the evaluation are the different node types, discussed in Section 7.1.2, the used medium, Section 7.1.4 and the evaluation plugins, Section 7.1.5. The nodes consist of several subcomponents, which are discussed in Section 7.1.3.

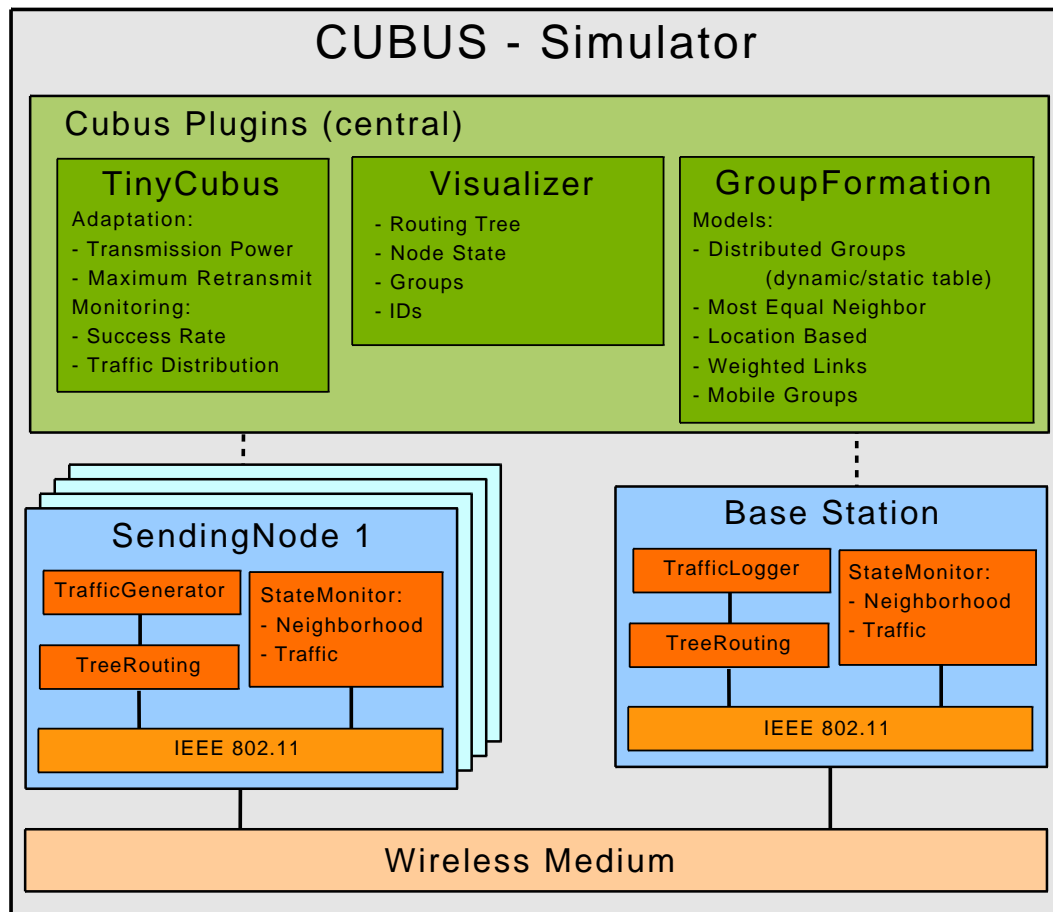


Figure 7.1: Scenario Evaluation Components

7.1.1 Simulator

The evaluation of algorithms in sensor networks requires a parallel execution on many sensor nodes. In case of group formation models about a hundred nodes are needed for evaluation, which requires the usage of a network simulator. Using a simulator has the advantage that a lot of different configurations can be automatically simulated.

Finding optimal adaptation settings for nodes in sensor networks requires a lot of simulations, because of the large parameter space. Since the *CUBUS* simulator [CUB07] is a fast network simulator, because the algorithms are executed natively on the host, this simulator is used during the evaluation. Another feature offering the usage of this simulator is the powerful scenario specification language.

7.1.2 Node Types

During the evaluation two different node types are used. The first type, called *Sending Node*, simulates a node collecting data about its environment. Since the collected data is needed outside the sensor network, this data has to be sent to the second type of nodes, the *Base Station*, which simulates the gateway to another network, where the data is required.

Generally these two nodes are using the same components for sending data through the network. These components are:

- *IEEE80211*: Low level protocol for data transmission (part of the *CUBUS* simulator)
- *TreeRouting*: Implementation of a routing protocol
- *Statemonitor*: Component to collect information used by the *GroupFormation* and *Visualizer*

Depending on the type of node, the last component is a *TrafficGenerator* or *TrafficLogger*. A detailed description of these components is topic of the next section.

7.1.3 Node Components

In the previous section the two different node types, used in the scenarios, were introduced. Since these nodes are built up by several subcomponents, this section introduces the used components and their configuration. The scenarios, discussed in Section 7.2, use these components with the introduced configuration.

7.1.3.1 TrafficGenerator

A sensor network in real applications is often used to observe the environment and retrieve data about it. The scenarios used for the evaluation of models simulate such an application and, therefore, the *Sending Nodes* have a *TrafficGenerator* on top of the protocol stack. The *TrafficGenerator* creates randomly data packets and sends them down the protocol stack.

The size of the data packets is constantly 30 bytes. The traffic generated by one node is on average three bytes per second, which results in sending one packet every ten seconds.

```

1 <item value="/TrafficGenerator">
2     <minPackageSize value="30" />
3     <maxPackageSize value="30" />
4     <byteTraffic value="3" />
5 </item>

```

Algorithm 7.1: TrafficGenerator Configuration

7.1.3.2 TrafficLogger

The target for the data, generated by the *Sending Nodes*, is the *Base Station*. To count the number of packets arriving at the *Base Station*, the *TrafficLogger* is used. This component waits for incoming data and counts the number of incoming bytes.

7.1.3.3 TreeRouting

The *TreeRouting* component implements a routing algorithm based on spanning trees. The root of that tree is the base station. Since every node knows its parent in the tree, the tree can be used to route the data to the base station using multihop communication.

Since there exist a lot of different implementations with various metrics, defining for example how to select a parent node, this section explains the used implementation in detail. The exact knowledge of the mode of operation is necessary to understand and interpret the results gained from simulating the scenarios.

Spanning Tree Construction

The construction of the spanning tree is provided by a distributed algorithm. This algorithm is executed on every node in the network. Algorithm 7.2 shows the construction process in pseudo code.

```

1  switch event do
2    case OnSendBeacon
3      beacon.set(myID);
4      beacon.set(myRootDistance);
5      beacon.set(myNeighborhood);
6      beacon.broadcast();
7    case OnReceiveBeacon
8      myNeighborhood.add(beacon.getID());
9      myNeighborhood.set(beacon.getID(), beacon.getRootDistance());
10     myNeighborhood.set(beacon.getID(), beacon.getRSSI());
11     myNeighborhood.refreshTTL(beacon.getID());
12     if beacon.getNeighborhood().contains(myID) then
13       myNeighborhood.markBidirectionalLink(beacon.getID());
14     end
15   case OnAllBeaconReceived
16     myNeighborhood.updateTTL();
17     parentCandidates = myNeighborhood.copy();
18     parentCandidates.removeNodesWithZeroTTL();
19     parentCandidates.removeNodesWithoutPathToRoot();
20     parentCandidates.removeNodesWithoutBidirectionalLink();
21     parentCandidates.removeNodesWithoutMinimalRootDistance();
22     parentCandidates.removeNodesWithoutMaximalRSSI();
23     myRootID = parentCandidates.getOldestNode().getID();
24     myRootDistance = parentCandidates.getOldestNode().getRootDistance() + 1;
25   end
26 end

```

Algorithm 7.2: Spanning Tree Construction

The algorithm is event driven, which means that all actions are caused by events. In case of the construction process there are three events: *OnSendBeacon*, *OnReceiveBeacon*, and *OnAllBeaconReceived*.

The *OnSendBeacon* event is triggered periodically, with the interval of three seconds. Whenever the event occurs, a beacon with the node ID, the distance to the base station and a list of neighborhood nodes is sent per broadcast. Since the space in the beacon is limited only ten neighborhood nodes are included. The selection of the ten nodes is based on the time, when the node was detected first time.

The occurrence of the event *OnReceiveBeacon* indicates the arrival of a beacon from another node. The sender of the beacon is included in the neighborhood list. To detect node failures or support moving nodes, the list implements the soft state approach. If no beacon of a node arrives for 30 seconds, then the node is removed from the list.

Since the transmission power of the nodes is adapted during the simulation, links between node are not necessarily bi-directional. To avoid situations where nodes select parents, without having a bi-directional link between them, a *BidirectionalLinkFlag* is introduced. If the neighborhood list of the incoming beacon contains the own ID, the link is assumed to be bi-directional.

The last event used for spanning tree construction is *OnAllBeaconReceived*. This event is triggered before the *OnSendBeacon* event. The first performed action is to update the time-to-live values of the neighborhood list, or in other words decrease them. A time-to-live value equal zero indicates a timeout. After that, the optimal parent node is searched.

The list of potential parents is limited step by step. After removing timeouted nodes and those without a path to the root, nodes which are connected with an uni-directional link are deleted. From the remaining nodes those with a minimal root distance are searched. In case of several nodes with the same distance, those nodes with a maximum *RSSI* value are selected. Theoretically this list can contain more than one node. The final selection is based on the time when the node was detected first.

Data Routing

Data, generated by a *TrafficGenerator* or received by any other node, is routed to the base station. Every node sends the data using a unicast packet to its parent node. In case of no available parent, no path to the base station is known and the data is discarded. After a short initialization phase, where the neighborhoods are exchanged, every node knows some neighbors with a route to the base station and, therefore, no data should be lost because of missing paths.

7.1.3.4 Data Link Layer

The communication between two neighboring nodes is controlled by the *IEEE80211* component, which is part of the *CUBUS* network simulator. Since unicast packets are transmitted using acknowledgments, lost packets are automatically retransmitted. The maximum number of retransmits is set to three.

7.1.3.5 Statemonitor

The last component used in the protocol stack is the *Statemonitor*. The functionality of this module is to collect the required information used for group formation. This information is:

- Calculation of the *Neighborhood Value*
- Counting the traffic of nodes
- Providing the location of nodes
- Storing the group ID

In contrast to the previous components, this one is only logically present. Logically means that the code for this component is integrated in other modules or is already provided by other modules. The reason for the integration is the fact that the *Statemonitor* uses mostly the same resources as the *TreeRouting* component. The integration avoids code duplication and the additionally included code in the *TreeRouting* module is very limited.

7.1.4 Medium

The nodes in the scenarios are interconnected by a wireless, shared medium. The bandwidth is set to 154kbps. The links between nodes are error free, which means that no bit errors are injected. Due to the shared medium, collisions between packets result in damaged data. The complete configuration of the medium is listed in Algorithm 7.3.

```

1 <item value="/Wireless">
2   <transmissionRate value="154" /> <!-- kbps -->
3   <propagationRate value="300000000" /> <!-- m/s -->
4   <errorModel value="/NoErrorModel" />
5   <signalPropagationModel value="/FreeSpaceModel" />
6   <signalDelayModel value="/DistanceDependentModel" />
7 </item>

```

Algorithm 7.3: Medium Configuration

7.1.5 Evaluation Plugins

Several plugins extend the functionality of the *CUBUS* simulator to evaluate the group formation models. These plugins implement the group formation algorithms, the adaptation engine and visualizer. In the following sections, the different modules are discussed in detail.

7.1.5.1 TinyCubus

The module *TinyCubus* simulates the behavior of the middleware *TinyCubus* [MLM⁺05]. It is used to adapt the transmission power of the nodes, by taking the formed groups into account.

To support the evaluation process the *TinyCubus* component implements metrics to rate the quality of the used configuration. Using those metrics the module is able to perform different methods to adapt the nodes. The methods are:

- **AdaptNodesNo:** This method configures every node with the same setting, which is used to evaluate the benefits of group formation. No group formation is used.
- **AdaptNodesRandom:** The nodes are configured with random settings, which allows to find the optimal setting manually.
- **AdaptNodesGenetic:** The genetic method implements an automatic approach to find the optimal configuration by using *Hill Climbing* technique.
- **AdaptNodesAll:** This method starts a *Brute Force* approach to find the optimal configuration by simulating all possible configurations.

Due to the large parameter space it is possible to fix several node or group configurations and only adapt parts of the nodes.

7.1.5.2 Visualizer

The *Visualizer* is used to create graphical outputs of the network situation. The implemented methods are:

- *CreateScenarioIDs*: Creates an image of the network with all nodes and their IDs.
- *CreateScenarioTree*: Visualizes the routing tree created by the *TreeRouting* module.
- *CreateScenarioTraffic*: Creates an image of the network with all nodes and their traffic.
- *CreateScenarioGroups*: The formed groups are visualized by this method.
- *CreateScenarioRange*: This method illustrates the nodes and their transmission ranges.

7.1.5.3 GroupFormation

The group formation models are implemented in this method. Since the focus of the thesis is on the quality of the models, all models are implemented centrally. Due to the usage of a simulator it is possible to access all node attributes without sending additional messages, which may influence the simulation results.

The implemented models are:

- *Distributed Groups Static Table* using neighborhood metrics.
- *Distributed Groups Dynamic Table* using neighborhood and traffic metrics.
- *Most Equal Neighbor* using neighborhood and traffic metrics.
- *Location Based* using hexagon and circular patterns.

The Most Equal Neighbor model uses the introduced optimization, by using not only the best neighbor, but also the second best, if the difference between them is smaller than 20 percent.

7.2 Scenarios

The evaluation of the models requires scenarios. These scenarios copy real world situations and abstract them to the relevant attributes. As introduced in Chapter 4 there are two different topology structures, random and regular. Since the used metrics of the designed models are based on the location, traffic and neighborhood of nodes, the used scenarios have to cover that.

Models based on the location of nodes can be evaluated with all scenarios, because scenarios where the location of all nodes is equal do not occur in reality. A similar argument can be applied on scenarios where all nodes have the same traffic. The only situation that fulfills this demand is a scenario, where every node can communicate with every other node directly. Such a scenario also hardly has a real world equivalent, except in networks with only few nodes.

Based on these arguments the different models can be evaluated using scenarios with regular and random node distribution. Such scenarios allow evaluating models using the neighborhood of nodes, and also those using traffic or location. Random node distribution results in a different neighborhood size, while regular distribution results in equal sized neighborhoods.

In the following sections the different scenarios are discussed in detail. These are *Grid Pattern* with a regular node distribution and *Random Location* with random node distribution.

7.2.1 Grid Pattern

The first scenario used during the evaluation of the group formation models is the *Grid Pattern* scenario. The scenario is built up by 100 nodes. These nodes are arranged regularly on a grid with an edge length of ten nodes. As depicted in Figure 7.2 (a) the distance between the nodes is 30 meters.

In the scenario two types of nodes are used, which are introduced in Section 7.1.2. There are 99 *Sending Nodes* creating traffic and routing the data to the *Base Station*, which is located in the top left corner.

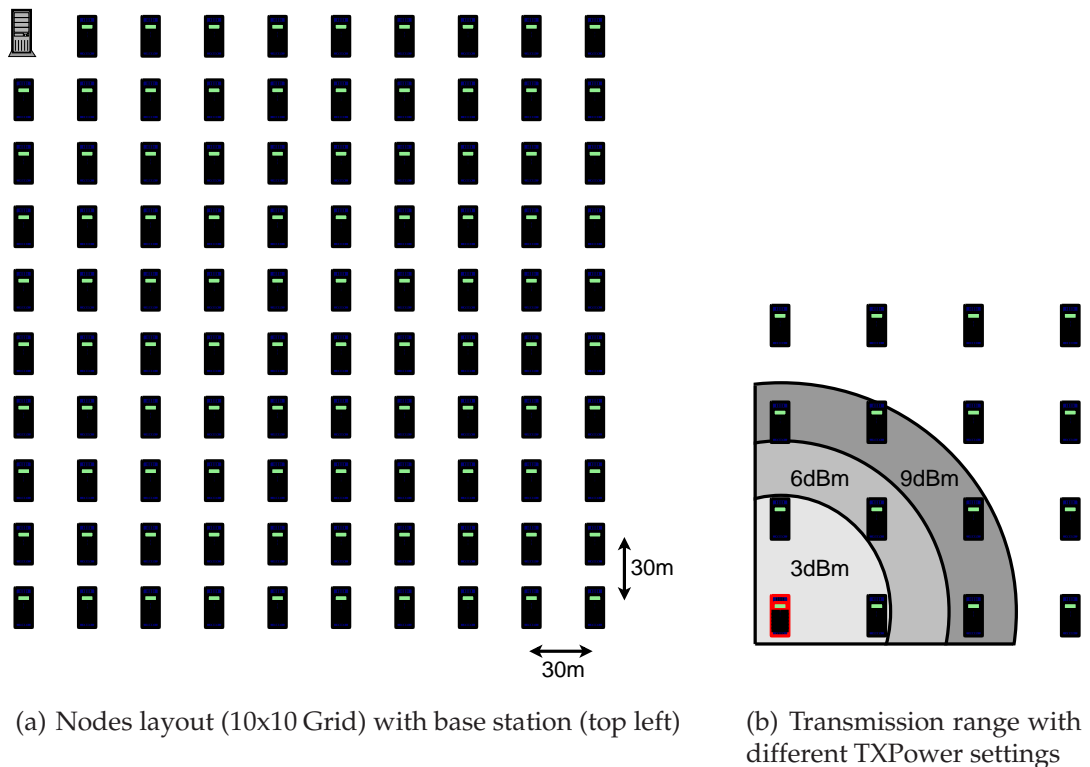


Figure 7.2: *Grid Pattern* scenario

Due to the regular node deployment, the neighborhood size is almost equal at each node. Only the nodes at the border have different neighborhood sizes. The benefit of such a scenario is the fact that influences based on the node density can be eliminated and, therefore, the effects resulting from traffic or location of nodes can be evaluated.

Since the neighborhood size depends on the transmission power of nodes it is also possible to evaluate the effects of large or small neighborhood sizes. Especially the correlation of different neighborhood sizes with different traffic patterns and the resulting throughput can be evaluated.

During the simulation of this scenario the nodes are grouped using the developed group formation models and each group is configured with one of the three transmission power settings. These settings are $3dBm$, $6dBm$ and $9dBm$ which results in transmission ranges of 39m, 55m and 79m respectively. Figure 7.2 (b) visualizes the neighborhood of a node sending with the three settings.

With the lowest setting only the four nearest nodes can be reached. using $6dBm$ it is possible to communicate with all eight surrounding nodes, or in other words it is possible to use diagonal links. The maximum transmission power allows to extend the neighborhood to 20 nodes.

The course of the scenario is structured as follows:

- **Initialization Phase** (0s – 300s): The routing algorithm constructs a routing tree. The *TrafficGenerators* create data and send them to the base station. Every node is configured with a transmission power of $6dBm$.
- **Group Formation Phase** (300s): The group formation component creates the groups using information gained from the *Statemonitor*. The information is collected during the *Initialization Phase*.
- **Reconfiguration Phase** (300s): The *TinyCubus* component reconfigures the nodes transmission power using the formed groups.
- **Adaptation Phase** (300s – 600s): During this phase the routing tree is adapted to the new situation. Due to the adapted transmission power, new neighborhoods are formed. The *TrafficGenerators* are disabled during this phase.
- **Test Phase** (600s – 1200s): After enabling the *TrafficGenerators* data is routed to the base station.
- **Evaluation Phase** (1200s): During this phase the total amount of data, received by the base station, is calculated. The quotient of received and generated data is named the *Target Ratio*, which is used to rate the quality of the network configuration (transmission power).

7.2.2 Random Location

The second used scenario is called *Random Location*. The setup consists of 70 nodes, randomly distributed on a $250m * 250m$ area. The position of the nodes is visualized in Figure 7.3 and listed in Algorithm C.1 in the appendix.

Due to the random deployment the sizes of nodes' neighborhoods vary and, therefore, the introduced *Neighborhood Value*, see Section 5.2.3, can be used to group nodes belonging to areas with an equal density. This allows an evaluation of models, depending on the *Neighborhood Value*, too.

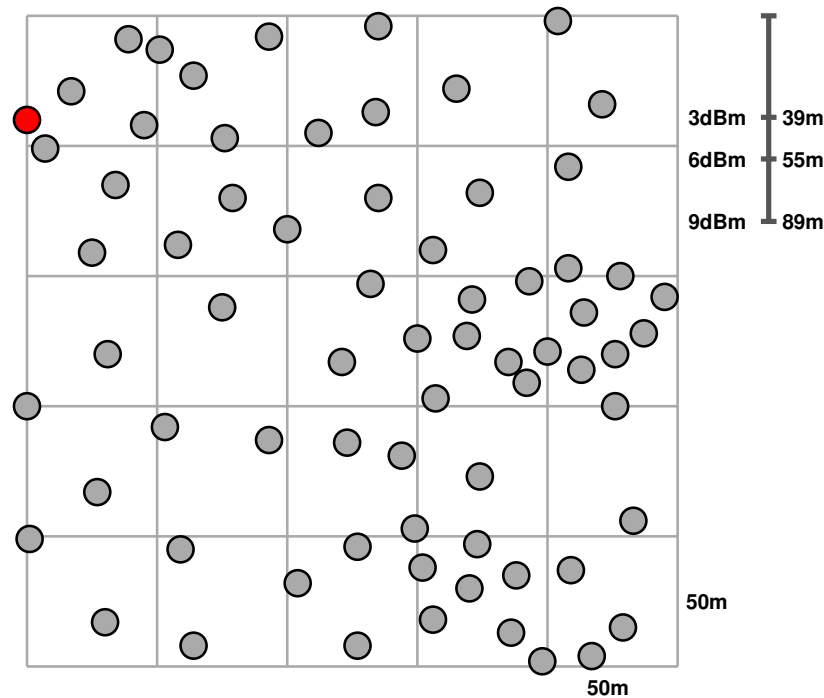


Figure 7.3: *Random Location* scenario: Node deployment and transmission ranges

To get an impression of the distances in the network, the key in the top right corner of Figure 7.3 shows the possible transmission ranges for different power values. When using the maximum transmission power of 9dBm , it is possible to communicate with nodes over a distance of 89m . The medium value of 6dBm allows a sending range of 55m , while the lowest setting joins nodes over a distance of 39m .

7.3 Measurement Results

In the following sections the quality of the group formation models are evaluated. Therefore the scenarios discussed in the previous section are simulated with differently grouped nodes and settings. The course of the evaluation is structured as follows:

1. **No Groups** Section 7.3.1: All nodes are configured with equal settings.
2. **Optimal Groups** Section 7.3.2: All nodes are configured with individual settings.
3. **Formed Groups** Section 7.3.5: Nodes are grouped using the models and configured with optimal settings.

The first step, using no groups, allows to identify the actual state without group formation models. Since every model introduces some overhead, messages to be transmitted or CPU time, the introduction of the models has to improve the network performance because otherwise the effort does not pay. When configuring every formed group with equal parameters, no differences between using and not using groups are visible. Based on that statement, the performance of grouped and optimally configured networks is higher than using no groups and, because of this, using no groups acts as the lower bound.

Since the first step identifies the lower bound, the second step is used to determine the upper bound. Every node is configured individually or in other words a model is used which assigns only one node to a group. Due to the individual configuration of every node, it is possible to get the same performance as using groups. In other words, the group-based configured network can be mapped to a network with individual configurations and, therefore, the performance of the node-based configuration acts as the upper bound.

Using the network simulator and the *AdaptNodesGenetic* functionality of the *TinyCubus* component, the optimal configuration for the network is searched. The behavior of this method is to randomly change the configuration of some nodes and depending on a performance improvement the new configuration is used. By iterating this step, the optimal setting of the network can be found. The described process is supported by manually changing or temporarily fixing the configuration of some nodes to get a faster convergence. Based on the optimal configuration, it is possible to determine optimal groups, because an optimal model groups exactly those nodes with equal settings.

In the following step, the developed group formation models are applied and groups are created. These groups are theoretically analyzed by comparing the formed groups with those generated in the previous step. Based on this, the quality of the models is roughly estimated. A detailed rating of the quality is determined by simulating the scenarios with groups formed by the models. The investigation of the optimal configuration for the formed groups of each model is performed by using the discussed process and the *AdaptNodesGenetic* method.

The quality of a network configuration is based on three factors:

- **Target Ratio:** Percentage of sensor values which are transferred to the base station. Using this value it is possible to calculate the *message loss* ($100 - TargetRatio$).
- **Variation:** A sensor network with a specific configuration should result theoretically always in the same *Target Ratio*. Since the link layer protocols are using random wait times to avoid collisions, there always exists a variation in the results. Since the *Target Ratio* is user input for the *TinyCubus* component, good models should result in a low variation.
- **Traffic:** The transmission of messages in the network costs energy. Since the reception of messages also costs energy, sending with a high transmission power should be avoided. To estimate energy costs of the different configurations, the *Traffic* metrics is used. The metrics is calculated by Formula 7.1. The received and transmitted data of all nodes is summed up and divided by the simulation time (see *Test Phase* in Section 7.2). During the search for an optimal configuration, the focus lies on maximizing the *Target Ratio* and minimizing the *Variation*. Since minimizing the required traffic is not a primary objective during the search of an optimal configuration, this metrics should be interpreted with caution. In situations where two models achieve equal results in the first two categories, the required traffic should be used for comparison. Another important aspect is the fact that using individually configured nodes, it is always possible to get an equal or better result than using any model.

$$traffic(scenario, nodes) \rightarrow \frac{\sum_{i \in nodes.count} (nodes_i.tx + nodes_i.rx)}{scenario.duration} \quad (7.1)$$

7.3.1 No Groups - Equal Settings

This section covers the results for equally configured nodes for both scenarios, *Grid Pattern* and *Random Location*.

7.3.1.1 Grid Pattern

The differences of the three configurations concerning the average *Target Ratio* are minimal, see Figures 7.4 (a), 7.5 (a) and 7.6 (a). The values are all between 51% and 55%. When comparing the variation of the values the differences are enormous. While the configurations *6dBm* and *9dBm* result in a variation of about ± 10 , the *3dBm* configuration has a variation of ± 25 . Such a huge variation makes a quality of service guaranty infeasible.

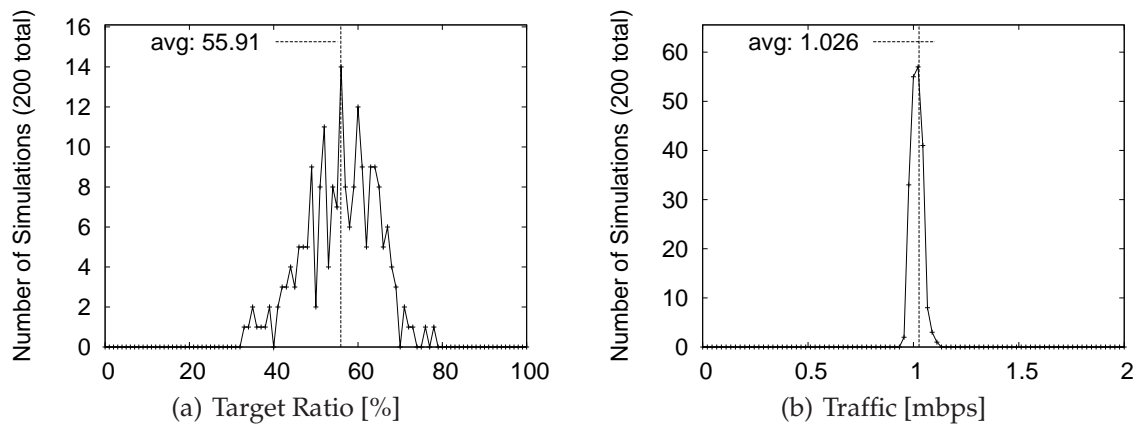


Figure 7.4: 10x10 nodes with same configuration (TXPower 3dBm)

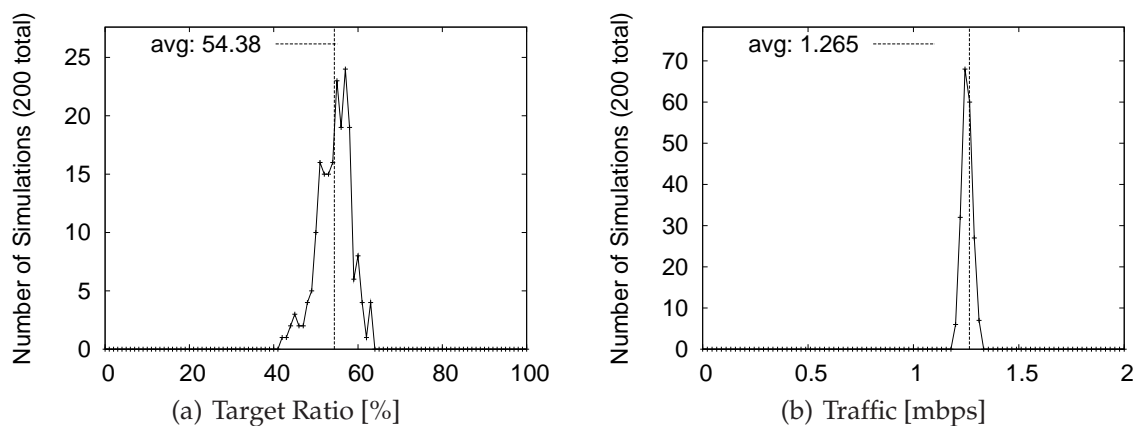


Figure 7.5: 10x10 nodes with same configuration (TXPower 6dBm)

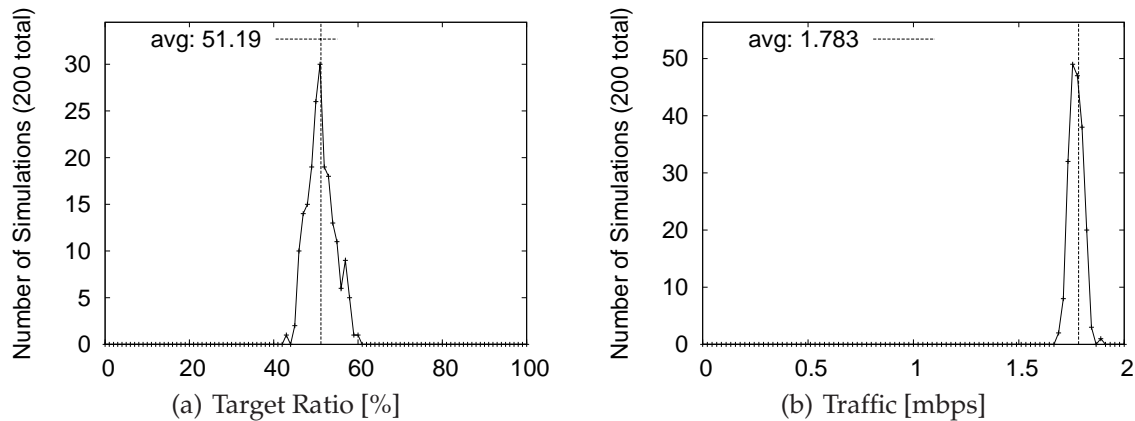


Figure 7.6: 10x10 nodes with same configuration (TXPower 9dBm)

The required traffic of the different configurations is as expected, see Figures 7.4 (b), 7.5 (b) and 7.6 (b). The traffic grows with the used transmission power. The lower number of hops cannot outweigh the large neighborhood of nodes and, therefore, the traffic doubles from 1.026mbps to 1.783mbps.

Based on these metrics all nodes should be configured with a transmission power of 6dBm. The Appendix A contains figures showing the number of transmitted kilobytes for each node.

7.3.1.2 Random Location

Analog to the *Grid Pattern* scenario, the nodes in the *Random Location* scenario are configured with equal settings. In contrast to the previous scenario, the differences between the three settings are huge. Using the minimal transmission power of 3dBm results in an average *Target Ratio* of 53%, see Figure 7.7 (a). The *Target Ratio* grows to 60% when using 9dBm, Figure 7.9 (a), and reaches its maximum of about 70% with a transmission power of 6dBm, Figure 7.8 (a).

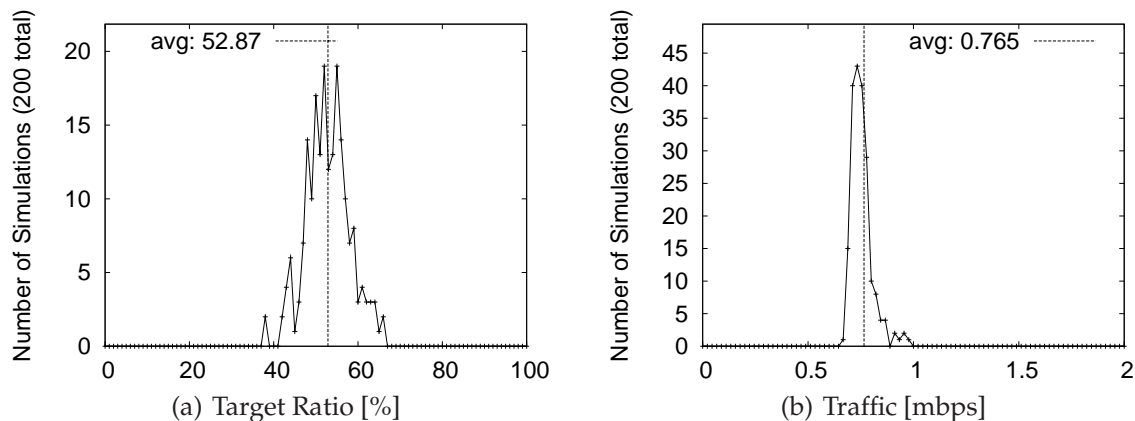


Figure 7.7: Random placed nodes with same configuration (TXPower 3dBm)

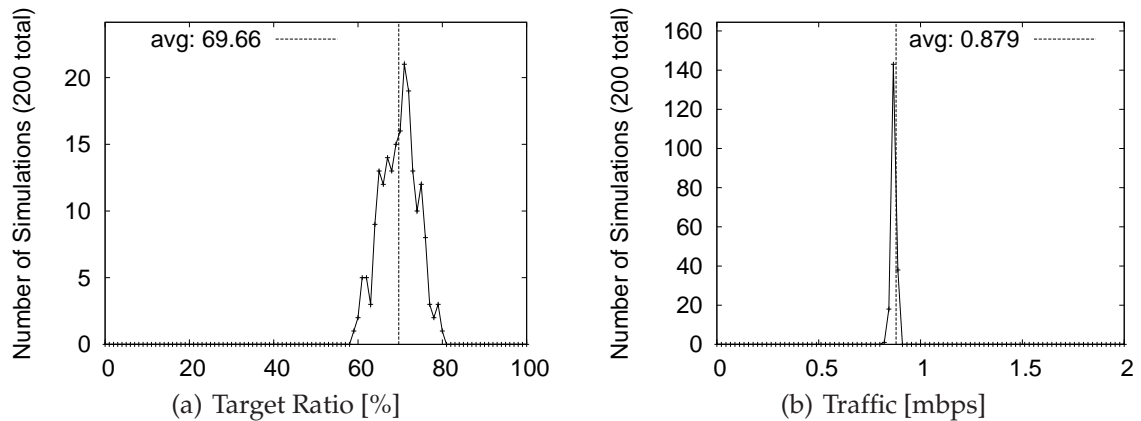


Figure 7.8: Random placed nodes with same configuration (TXPower 6dBm)

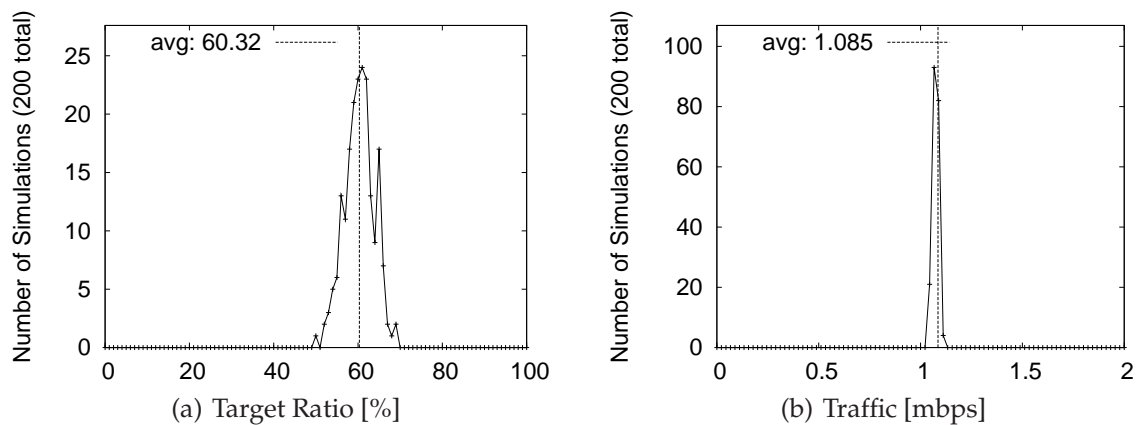


Figure 7.9: Random placed nodes with same configuration (TXPower 9dBm)

Like in the *Grid Pattern* scenario, a small transmission power results in the highest variation. The variation decreases with a growing transmission power. In contrast to the first scenario differences in the variation are not so significant. A transmission power of 3dBm results in ± 15 , while 9dBm results in ± 10 .

Figures 7.7 (b), 7.8 (b) and 7.9 (b), show the required traffic. Since the neighborhood of nodes grows with the transmission power an increasing traffic is expected. In contrast to the *Grid Pattern* scenario the effect of shorter paths to the base station can countervail the larger neighborhoods. In the *Random Location* scenario the required traffic grows from 0.765mbps to 1.085mbps.

Due to the extreme differences in the *Target Ratio* the optimal setting of equally configured nodes is a transmission power of 6dBm. The Appendix A contains figures showing the number of transmitted kilobytes for each node.

7.3.2 Optimal Groups - Individual Settings

The evaluation of the group formation models requires the knowledge of optimal groups or optimal configured nodes. Based on this knowledge, the formed groups can be compared to the optimal groups and the quality of the models can be rated. This section discusses the optimal configuration for the nodes.

7.3.2.1 Grid Pattern

Figure 7.10 (a) shows the optimal configuration for the nodes in the *Grid Pattern* scenario. The sensor network can be roughly divided into two areas. The first area contains the center and bottom right part, while the second area contains the nodes around the base station and the top and right border.

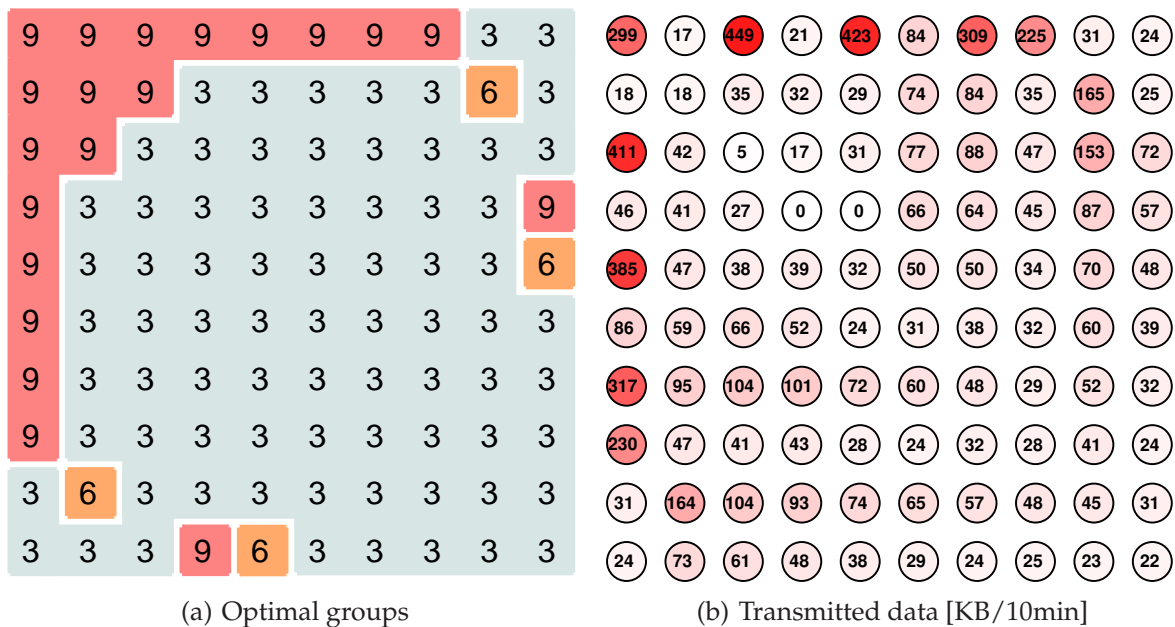


Figure 7.10: Optimal configuration for 10x10 nodes

A detailed analysis shows that these areas can be subdivided further into several groups.

The first group covers the nodes at edges of the network. The configuration of these nodes is most important, because the configuration must ensure that all traffic is routed along these nodes. Since a configuration of a transmission power of $9dBm$ reduces the number of hops, this setting is optimal.

The second group are the nodes around the base station. All nodes that are able to hear the base station beacons should be configured with $9dBm$, to allow a direct communication.

The third group covers those nodes in the center. To allow data routing using the edge nodes, the center has to send its data to the edges. Due to the routing metrics whose primary goal is to create short paths, the center must not send diagonally. Diagonal links result

in short paths to the base station without using the edges. Therefore, the center has to be configured with a transmission power of $3dBm$.

The last group is formed by the nodes at the bottom and the right edge. The configuration of these groups affects the *Target Ratio* only marginal. The differences between the optimal configuration and configure all nodes of this group with a transmission power of $9dBm$ is about 2 percentage points.

When analyzing the traffic flow, see Figure 7.10 (b), using the optimal configuration, it becomes obvious that all data is routed in straight lines to the edges. The nodes at the edges send their data to the base station. Due to the high transmission power of these nodes, large hops are possible.

The introduced configuration results in a *Target Ratio* of 84.5%, Figure 7.11. Comparing this value with the results of equally configured nodes (55% *Target Ratio*), the potential of introducing groups is evident. The variation of the results is ± 7 .

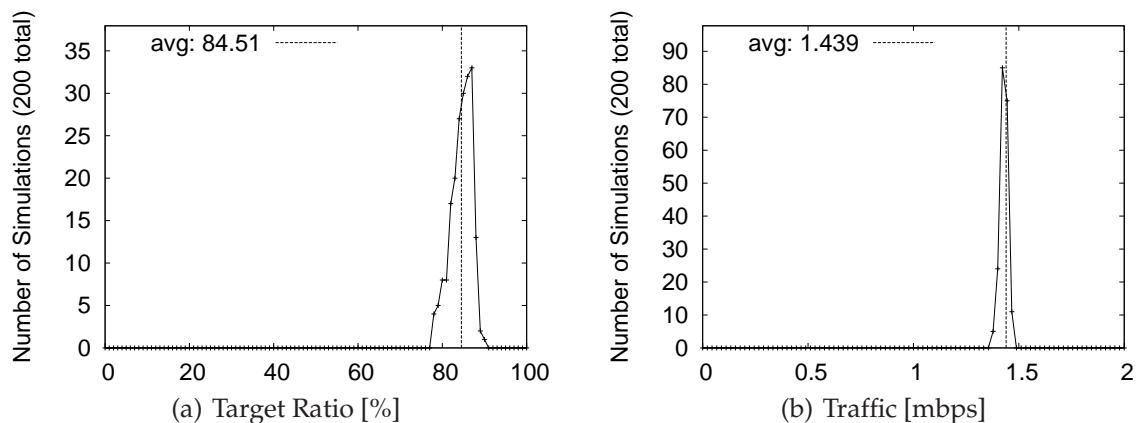


Figure 7.11: 10x10 nodes with individual configuration

The required traffic is $1.439mbps$. This value is between the requirements of equally configured nodes with a transmission power of $6dBm$ and $9dBm$. Due to the fact that most nodes are configured with a power value of $3dBm$, such a high value is surprising. The high value can be explained, because most data is transmitted by nodes with a high transmission power and, therefore, many nodes overhear the traffic, even if they are configured with a power value of $3dBm$.

7.3.2.2 Random Location

The nodes in the *Random Location* scenario can be divided into three groups. The first group covers those nodes around the base station, where the density is medium. The second group is build up by the nodes in the top right and bottom left corner with a low density. The nodes at the bottom right corner have the highest density and form the third group.

Figure 7.12 (a) shows the optimal configuration for this scenario. The nodes with the highest density are configured with a transmission power of $3dBm$. Due to the small size of these groups with a high density, about $50m \times 100m$, these nodes can communicate with a node outside this group using one hop. Nodes in a medium or a low density area are configured with a power value of $6dBm$.

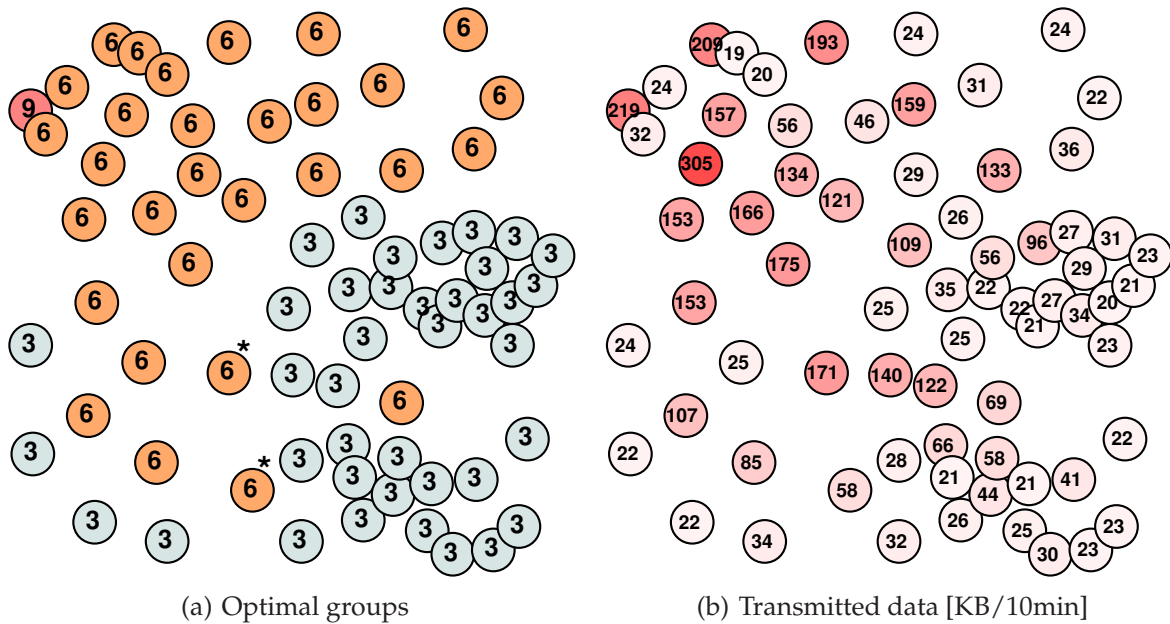


Figure 7.12: Optimal configuration for randomly placed nodes

The quality of a configuration is strongly related to the configuration of the two nodes (marked with * in Figure 7.12 (a)). These two nodes have to be configured with a value of $6dBm$. Such a configuration creates two additional data paths to the base station.

Figure 7.12 (b) shows the network with the nodes and the transmitted data. The figure shows that the data produced by the bottom part of the high density nodes is routed to the left side of the network. The data generated by the top part of the high density nodes is routed to the top of the network and then to the base station. In total, the traffic is equally distributed on four paths, which reduces the effort for the nodes on the paths.

The four nodes at the bottom left side are not used for data forwarding and, therefore, the optimal configuration is $3dBm$. A higher value increases the possibility of collisions and, because of this, reduces the performance of the routing path above these nodes.

Using the introduced configuration results in a *Target Ratio* of about 85%, see Figure 7.13 (a). The benefit compared to the equally configured nodes is only 15 percentage points, which limits the room for improvements of the group formation models.

The variation of the *Target Ratio* shrinks to ± 6 . When comparing this value to the variation resulting from equally configured nodes (± 10), the individually configured nodes produce more repeatable results.

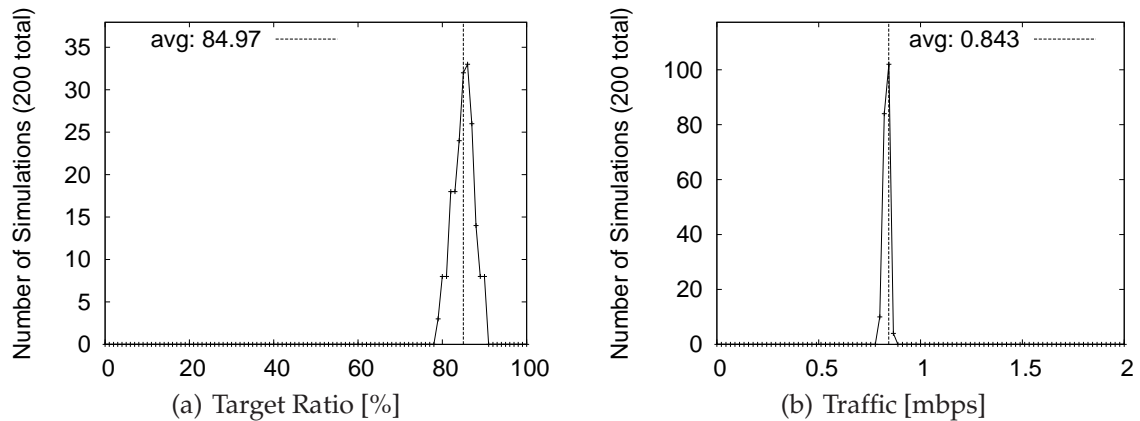


Figure 7.13: Randomly placed nodes with individual configuration

The traffic required by the optimal configuration is 0.843mbps , see Figure 7.13 (b). Equally configured nodes with a transmission power of 3dBm require 0.765mbps and 6dBm require 0.879 . Since half of the nodes are configured with 3dBm and the others with 6dBm , this result is as expected.

7.3.3 Group Formation Metrics

The group formation is based on several metrics: neighborhood, traffic and location. A general discussion of these metrics is provided in Chapter 4 and Section 6.1.2. Since the location of the nodes in the scenarios is discussed in Section 7.2, the following sections focus on the neighborhood and traffic metrics.

7.3.3.1 Grid Pattern

Figure 7.14 (b) shows the sensor network of the *Grid Pattern* scenario. For each node the number of neighbors is included. The transmission power of all nodes is 6dBm , which allows a vertical, horizontal, and diagonal communication. Based on the location of the nodes, theoretically three different values should exist.

The nodes in the corners can communicate with three neighbors and nodes at the edges have five neighbors. All remaining nodes in the center of the network have eight possible neighbors. Since the neighborhood detection is performed in parallel to the normal network activity, i.e. routing data to the base station, some beacon messages are lost and, therefore, the actual number of detected neighbors varies between six and eight.

The second used metrics is based on the traffic at the nodes. As discussed in Section 4.3.1, the sum of transmitted and received traffic is used. Figure 7.14 (b) shows nodes with the traffic. The unit of the given values is $\text{KB}/10\text{min}$. The snapshot shows the traffic distribution of the *Group Formation Phase*, see Section 7.2.1.

Due to the configured transmission power, the data is routed in straight lines to the diagonal axis (top-left to bottom-right). Data generated above the diagonal is routed horizontally and data below the diagonal is routed vertically.

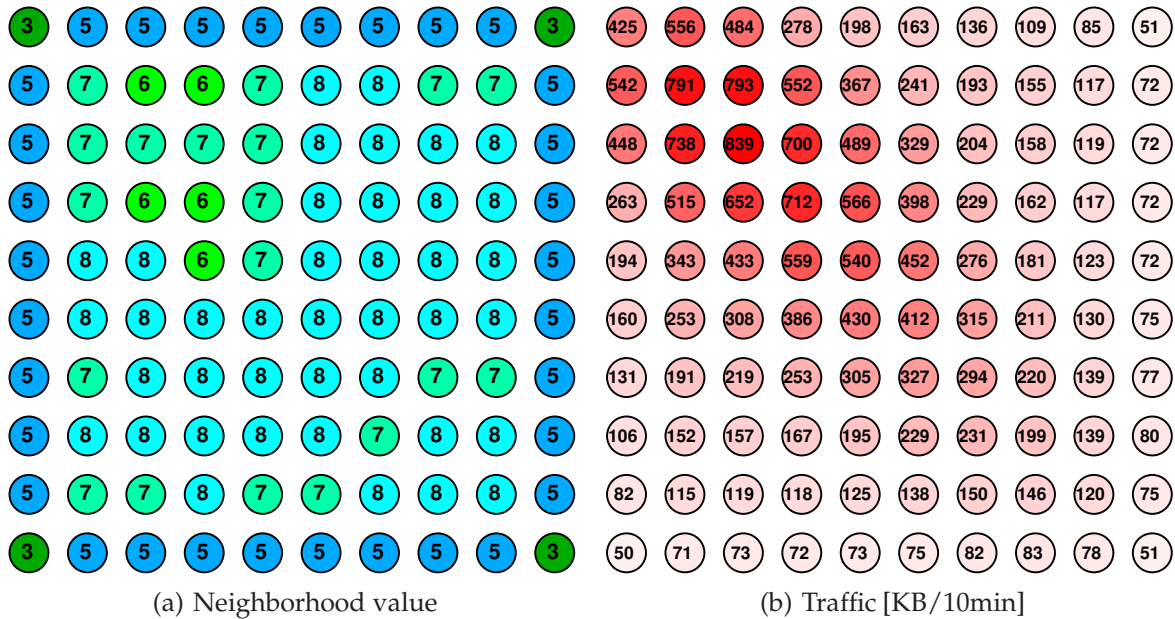


Figure 7.14: Group formation metrics for 10x10 Nodes with TXPower 6dBm

Based on the formed routing tree, nodes with most traffic are located near the diagonal and near the base station. With increasing distance to these nodes, the traffic decreases.

7.3.3.2 Random Location

Since the neighborhood metrics is based on the density of nodes, scenarios where nodes are randomly distributed are ideal for this metrics. Figure 7.15 (a) shows the networks of the *Random Location* scenario with the neighborhood value attached to each node.

Due to the different density distribution in the network, the size of the neighborhoods varies from two to 15 nodes and can be distinguished in three different groups. The nodes in areas with a low density (top-right and bottom-left corner) have values between two and six. In the area with a medium density (top-left corner) the neighborhoods have sizes between five and nine. The last group covers those nodes in a high density area (right side). The number of nodes in the neighborhoods varies from nine to 15.

The distribution of the traffic is very similar to the *Grid Pattern* scenario. Nodes near the base station have the maximum traffic. The traffic decreases with increasing distances to this group of nodes.

In areas with a high density, the traffic is noticeably higher than in areas with a low density. Like in the *Grid Pattern* scenario, the differences between the nodes are high. Nodes at the edges of the network have values of 40 – 70KB/10min and the nodes with maximum traffic have to handle a traffic of 600 – 650KB/10min

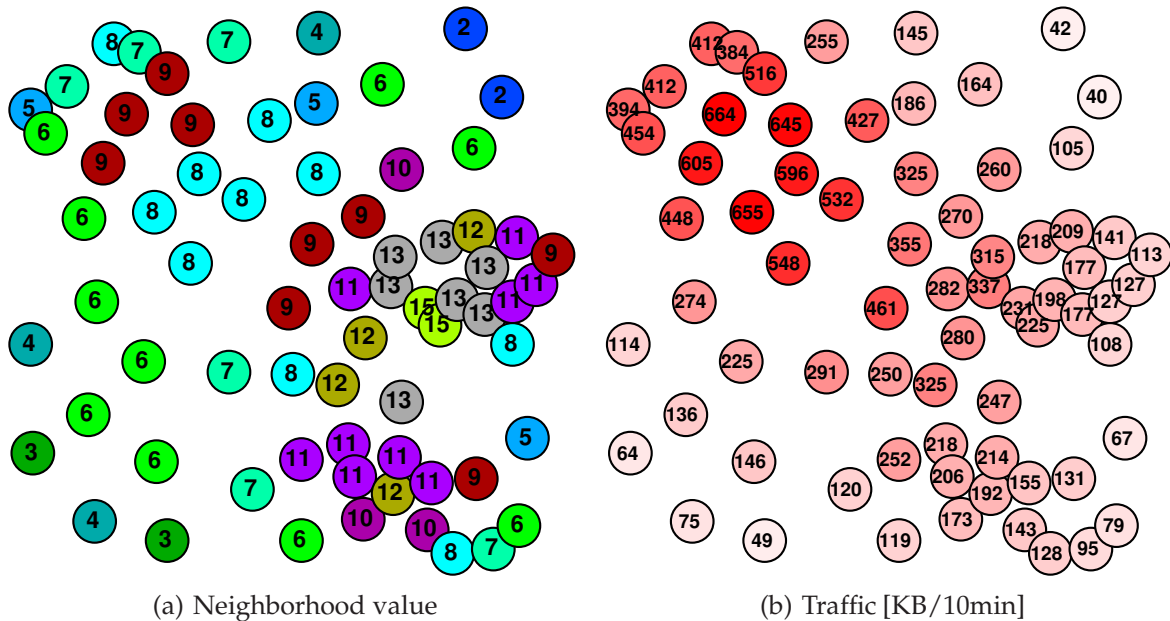


Figure 7.15: Group formation metrics for randomly placed nodes with TXPower 6dBm

7.3.4 Formed Groups

Based on the metrics discussed in the previous section, this section covers the formed groups. As introduced in Section 7.1.5.3, the nodes are grouped by the *Location Based*, *Most Equal Neighbor*, and *Distributed Groups* models. In the following sections for each model the formed groups are discussed.

7.3.4.1 Grid Pattern

This section covers the formed groups for the *Grid Pattern* scenario. Figure 7.16 (a) shows the groups based on the *Location Based - Circular* model. When comparing the group sizes with those of other models, it can be stated that group sizes are all about the same.

The second model based on the location of nodes is called *Location Based - Hexagon*. Figure 7.16 (b) visualizes the groups formed by this model. Comparing the groups with those of the *LB-C* model a difference concerning the group borders becomes visible. For example, the node in the third column and fifth row should be assigned to group five to correspond with the optimal group properties. Using the *LB-H* model results in several nodes which should be assigned to the neighboring group.

Figure 7.16 (c) shows the groups formed by the *Most Equal Neighbor - Neighborhood* model. Since this model is based on the neighborhood of the nodes and all nodes in the center have the same neighborhood, the groups are more or less randomly formed. The nodes at the edges are grouped, even though the set of nodes is split up in many small groups. By merging groups at the edges to one group and merging the groups in center to a second group, this model allows a distributed group formation based on the neighborhood of nodes.

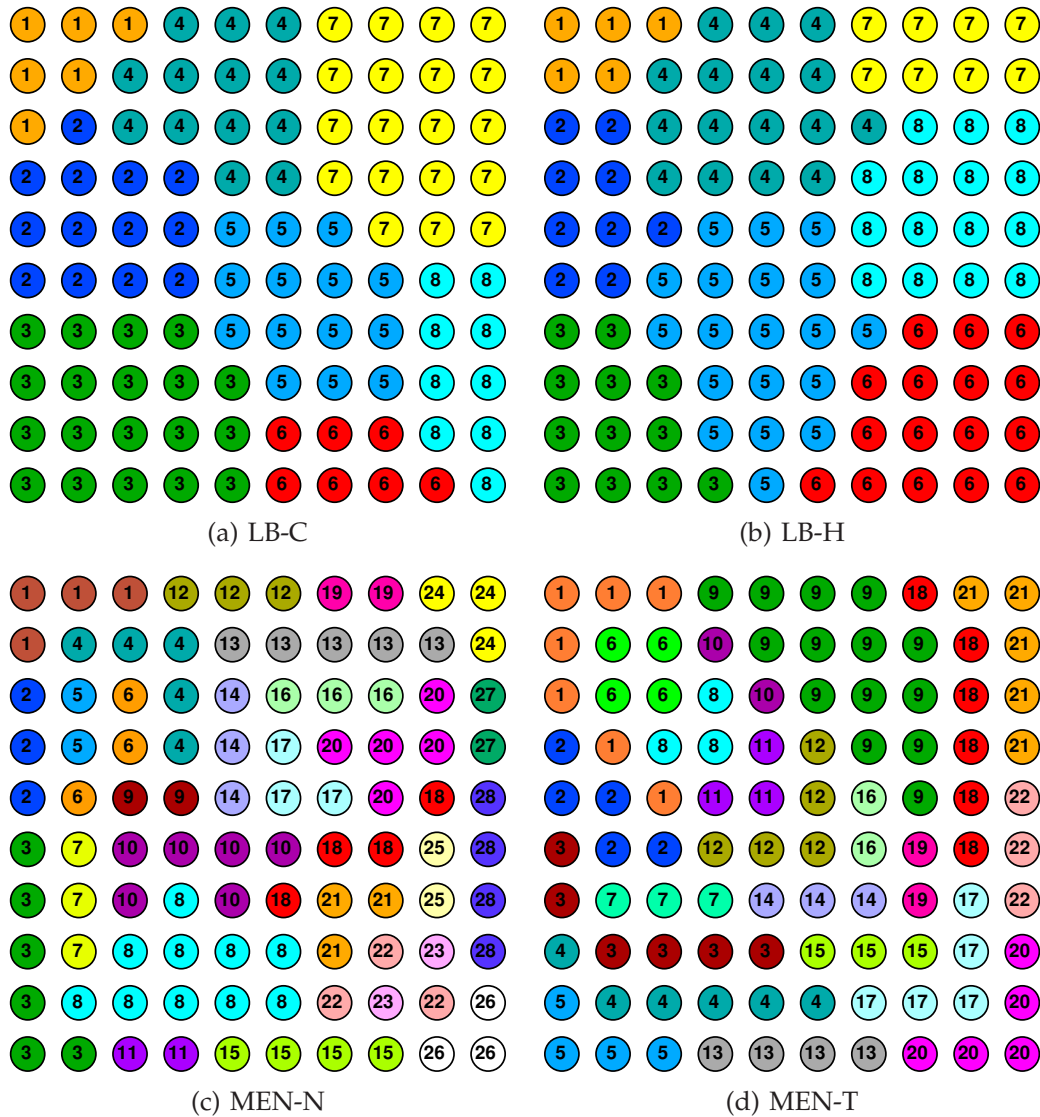


Figure 7.16: Grouped 10x10 nodes with model: LB-C, LB-H, MEN-N, MEN-T

The second distributed approach is using the traffic metrics. The groups formed by the *Most Equal Neighbor - Traffic* model are visualized in Figure 7.16 (d). When comparing the formed nodes with the traffic distribution, see Figure 7.14 (b), it becomes obvious that the model groups exactly those nodes with the same traffic.

The first model forming distributed groups is the *Distributed Groups Dynamic Table - Neighborhood*. Figure 7.17 (a) shows the groups formed by this model. Since the group formation is based on the neighborhood, this model forms two groups. The nodes in the center build the first and the edge nodes a second group. Due to the centralized coordination, it is possible to avoid the creation of several small groups.

In Figure 7.17 (b) the groups formed by the *Distributed Groups Dynamic Table - Traffic* model are visualized. When comparing the formed groups with those of the model *MEN-T*, see

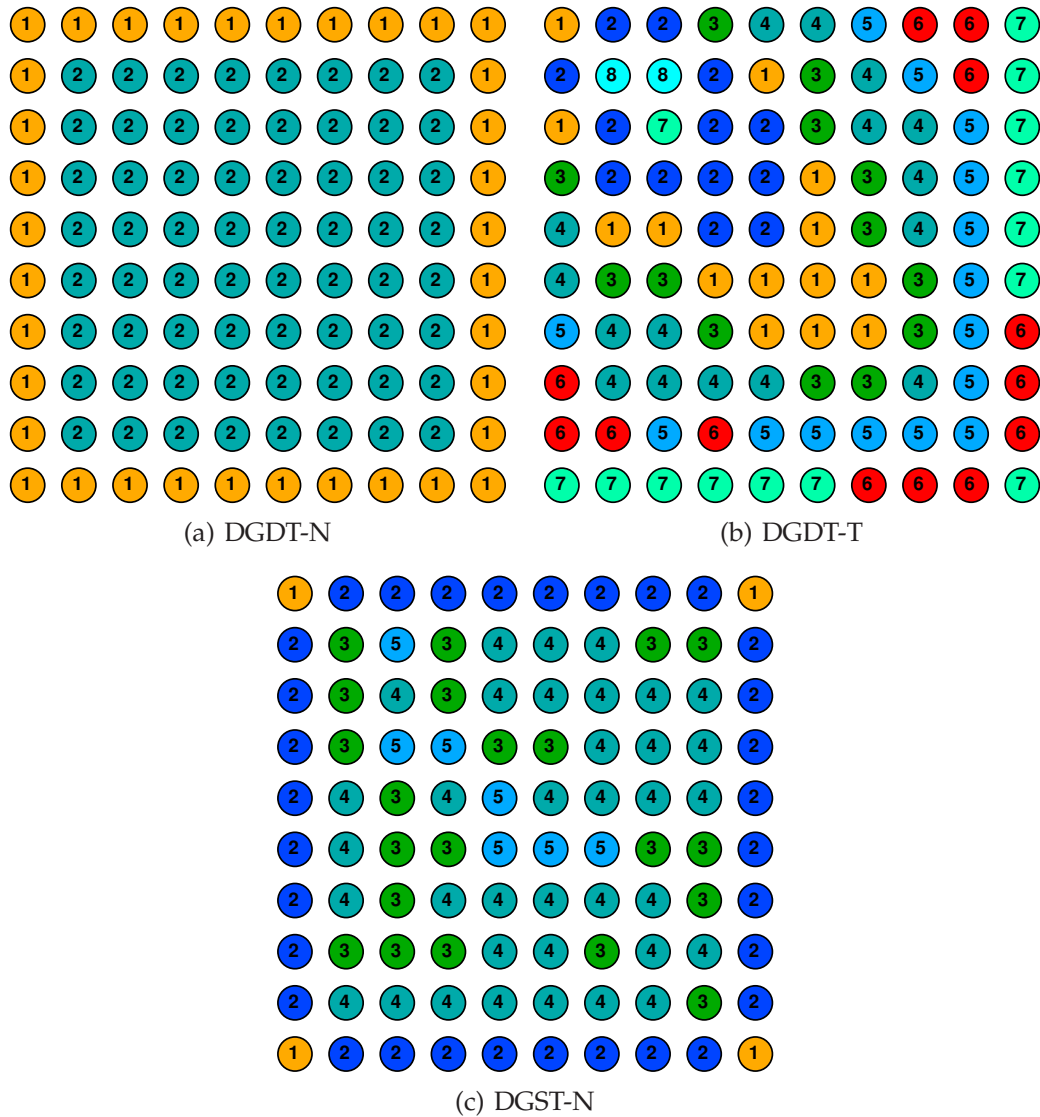


Figure 7.17: Grouped 10x10 nodes with model: DGDT-N, DGDT-T, DGST-N

Figure 7.16 (d), it becomes apparent that both models are forming nearly the same groups. The small difference is the fact, that the *MEN-T* model splits up the groups, formed by *DGDT-T*.

Since the groups formed by the *DGDT-N* and *DGDT-T* models are comparable to those of the *MEN-N* and *MEN-T* model, the quality of the formed groups is comparable, too. Although the differences between these two pairs of models have no effect on the quality of the models, the differences have a huge influence when using those models in real applications. In contrast to the *DGDT-N* and *DGDT-T* the models based on the idea of the most equal neighbor are completely distributed approaches and, therefore, the scalability of the *MEN-N* and *MEN-T* models is much higher.

The groups formed by the *Distributed Groups Static Table - Neighborhood*, see Figure 7.17 (c), are almost the same as those formed by the *DGDT-N* model. The *DGST-N* model divides the groups, created by the *DGDT-N* model, into smaller groups. Since the nodes are more or less randomly assigned to those groups, these groups are not stable. Stable means that these groups do not change every time the group formation process is activated. Due to the unstable groups and the random assignment a meaningful evaluation of the formed groups for this model is not possible. All groups in the center have to be configured with the same values, which results in the same groups that are formed by the *DGDT-N* approach. Since the *DGST-N* model is completely distributed and forms the same groups as the *DGDT-N* model, a real application should always prefer the *DGST-N*.

7.3.4.2 Random Location

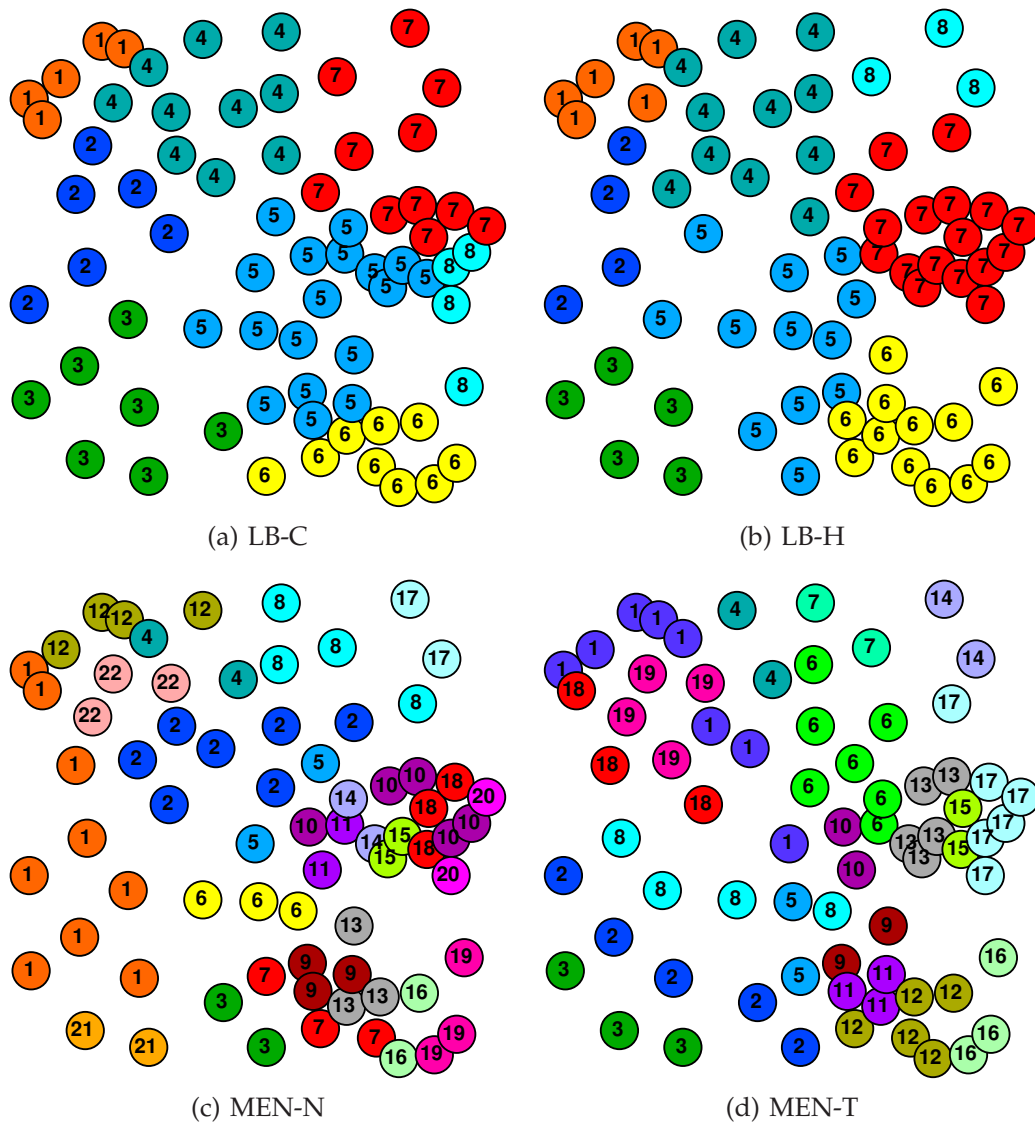


Figure 7.18: Grouped random located nodes with model: LB-C, LB-H, MEN-N, MEN-T

The formed groups of the *Random Location* scenario are introduced in this section. Figure 7.18 (a) shows the groups resulting from the *Location Based - Circular* model. Since the model only focuses on the location of the nodes, ignoring the neighborhood, some groups contain nodes with large and small neighborhoods. An example of such a group is group 7. The nodes at the top of this group are in an area with a small density and the nodes at the bottom have many neighbors.

In Figure 7.18 (b) the groups formed by the *Location Based - Hexagon* model are visualized. An examination of the borders between the groups highlights the design idea of this model. All borders between neighboring nodes consist of several nodes. The group sizes of both location dependent models are varying heavily. Since these sizes are resulting from the different densities in the network, the avoidance requires a density dependent distribution of the virtual cluster heads.

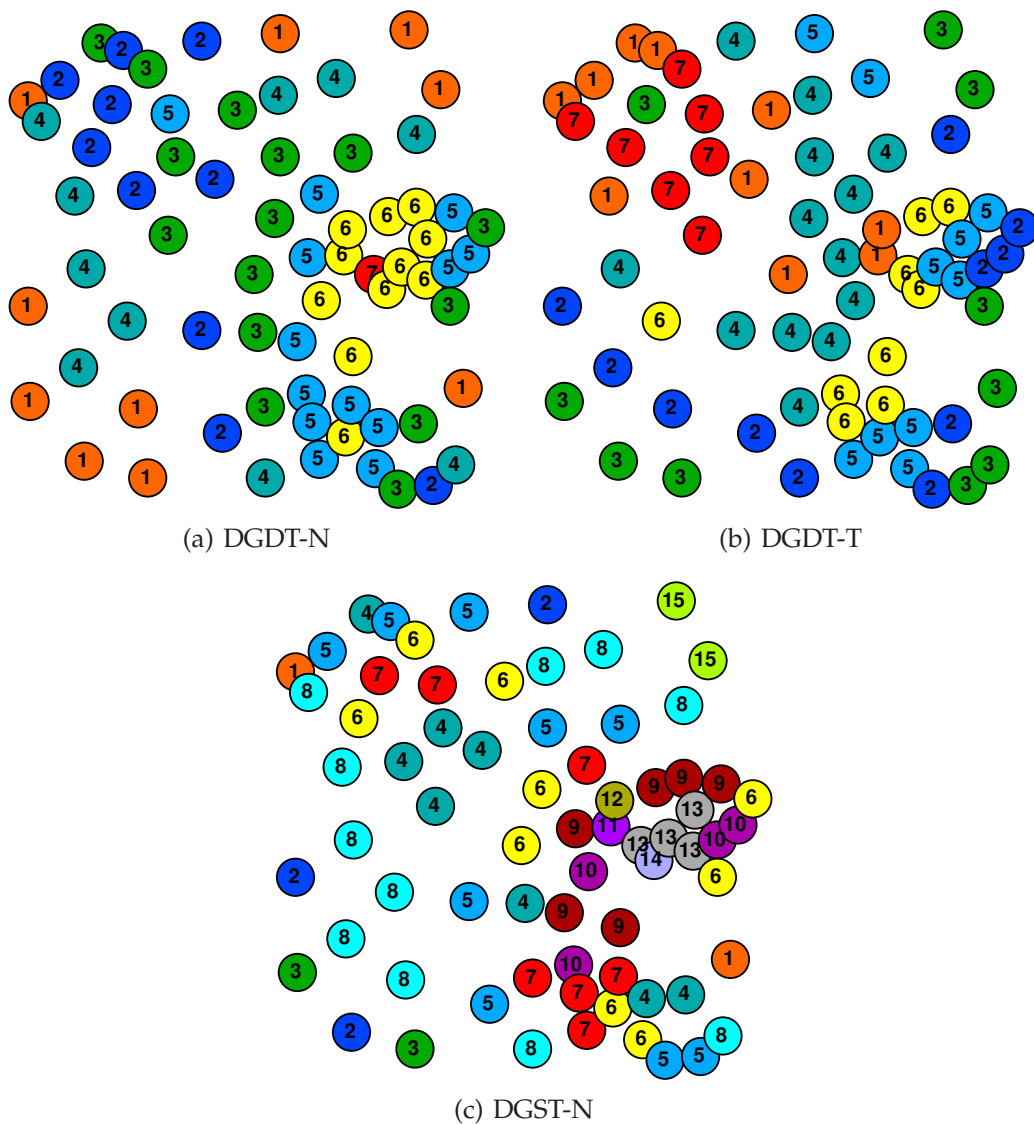


Figure 7.19: Grouped random located nodes with model: DGDT-N, DGDT-T, DGST-N

Figure 7.18 (c) shows group formation based on the neighborhood of nodes using the *Most Equal Neighbor - Neighborhood* model. As discussed during the introduction of the groups formed in the *Grid Pattern* scenario, this model forms a large number of groups. When merging some of these groups, for example 1 and 21 or 8 and 17, it becomes obvious that exactly those nodes with an equal neighborhood size are in the same group.

When using the *Most Equal Neighbor - Traffic* model the groups shown in Figure 7.18 (d) are formed. Keeping the traffic distribution in this scenario in mind, see Figure 7.15 (b), the formed groups match up with the traffic distribution. Due to the distributed creation, small groups are created.

The groups formed by the *Distributed Groups Dynamic Table - Neighborhood* are visualized in Figure 7.19 (a). When comparing these groups with those of the *MEN-N*, see Figure 7.18 (c), the similarities between both models become visible. Merging the groups 1, 8, 17, 21 (*MEN-N*), which are containing the nodes in low density areas, covers nearly the same nodes as the groups 1, 4 (*DGDT-N*).

An examination of the groups formed by the *Distributed Groups Dynamic Table - Traffic*, visualized in Figure 7.19 (b), results in similar statements as for the previous model. The groups 2, 3, 5 (*DGDT-T*) cover nearly the same nodes as groups 2, 3, 7, 16, 17 (*MEN-T*). As already mentioned during the discussion of the *Grid Pattern* scenario, the models based on the *Most Equal Neighbor* approach form the same groups as the models using distributed groups. The last model used to create groups is the *Distributed Groups Static Table - Neighborhood* model, see Figure 7.19 (c). As outlined during the discussion of this model in the *Grid Pattern* scenario, this model forms almost the same groups as the *DGDT-N* model. Due to the absence of a central coordinator the groups are smaller and the sizes of the groups are varying from one node minimum to eleven nodes at maximum.

7.3.5 Model Quality - Optimal Settings

This section covers the evaluation of the models' quality. The rating of the models is realized by configuring the network with the optimal setting for each group. In the following subsections the results for both scenarios grouped by the two location based and the models using the neighborhood and traffic metrics are shown. In other words the models *LB-C*, *LB-H*, *DGDT-N* and *DGDT-T* are evaluated. Due to the huge number of formed groups and the fact that merging some of these groups results in the groups formed by the *Distributed Groups Dynamic Table* model, an individual evaluation of the *Most Equal Neighbor* approach is not required. A similar argument can be applied to the *DGST* model. As discussed in Section 7.3.4 this model forms similar groups to the *DGDT* model and, therefore, no extra evaluation is needed.

7.3.5.1 Grid Pattern

The evaluation of the models using the *Grid Pattern* scenario is provided by this section. For each model the distribution of the *Target Ratio* and the required traffic is visualized as well as the data flow.

Location Based - Circular

Figure 7.20 (a) contains the *Target Ratio* distribution which emerges from grouping the network with the *Location Based - Circular* model and using the optimal configuration for each group. The *Target Ratio* is in average 72.88% (No Groups: 55%; Best: 84.5%). The network requires in average 1.238mbps traffic (No Groups: 1.265mbps; Best: 1.439mbps), see Figure 7.20 (b).

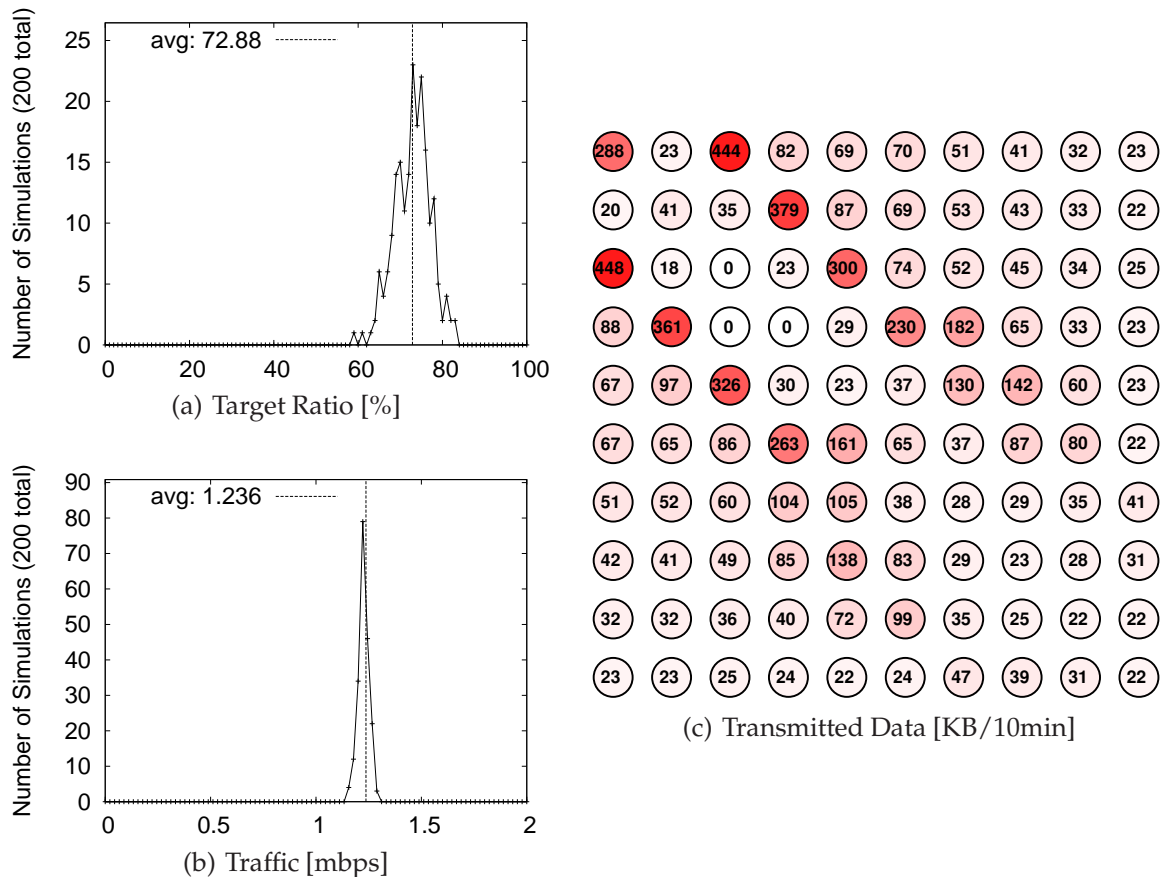


Figure 7.20: 10x10 nodes with model *LB-C* and optimal settings

The optimal setting ($txpower_i = \{9, 6, 3, 6, 3, 6, 3, 6\}$, transmission power values in *dbm* for group *i*, see Figure 7.16 (a) for group locations) configures the nodes around the base station (Group 1) with the maximum transmission power. Group 2 and 4, which are next to Group 1, are configured with a medium transmission power. The nodes with an even further distance to the base station (Group 3, 5 and 7) are set up with a minimal power value. The remaining nodes with a maximum distance (Group 6 and 8) are again configured with a medium transmission power. The configuration of the groups confirms the assumption, that the configuration is dependent on the distance to the base station.

When comparing the data flow of this model, which is visualized in Figure 7.20 (c), with the data flow of equally configured nodes, see Figure A.1 (b), only minimal differences are identifiable. Based on this observation, the target ratio enhancement is not the result of

different data routes, but the result of minimizing the number of collisions, which causes message loss.

Location Based - Hexagon

When grouping the nodes using the *Location Based - Hexagon* model and configuring each group with the optimal setting, the average *Target Ratio* is 60.03%, see Figure 7.21 (a). On the one hand this result is only marginal better as the one resulting from equally configured nodes (55%) and on the other hand the value has a high variation. Based on this variation, the required traffic varies, too (Figure 7.21 (b)). The average required traffic is with an average of 1.404mbps also noticeable higher than the traffic resulting from the *LB – C* model (1.238mbps).

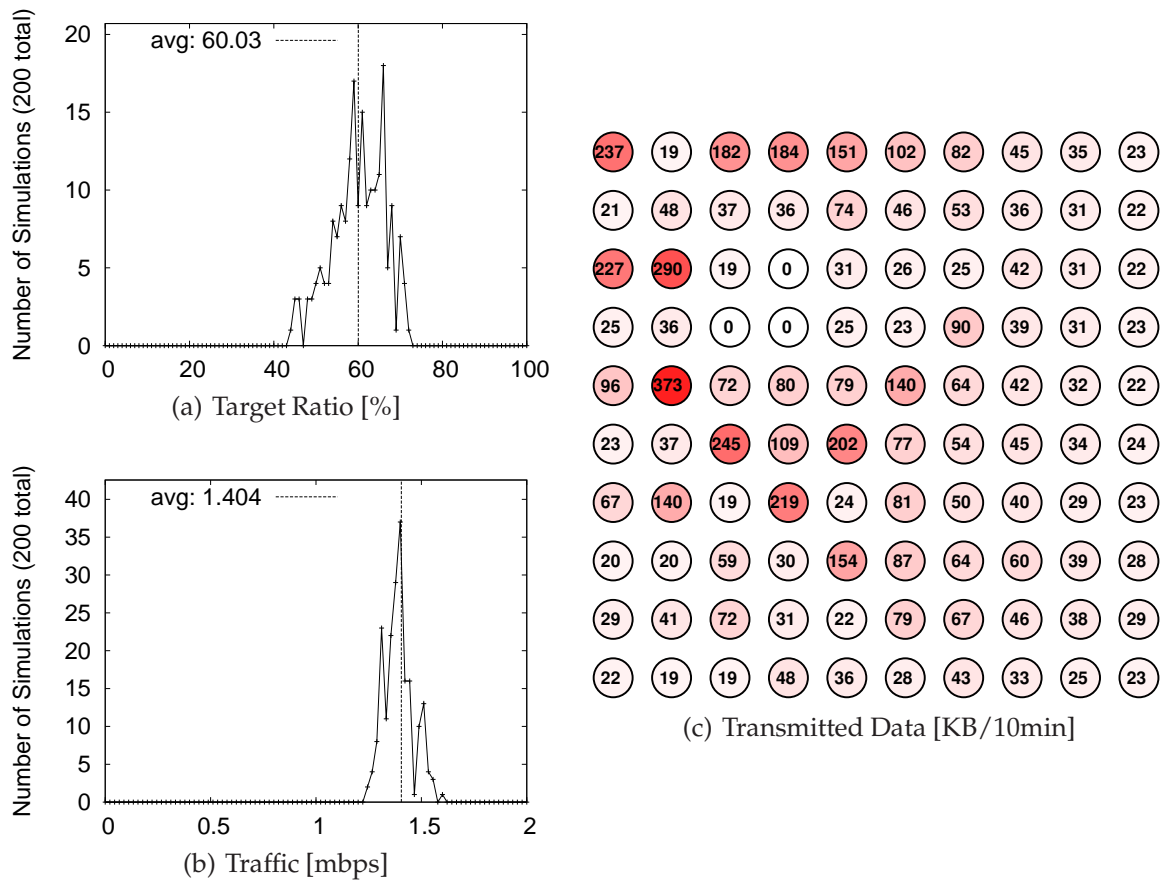


Figure 7.21: 10x10 nodes with model *LB-H* and optimal settings

When taking a closer look to the optimal configuration of the formed groups, the asymmetric settings are standing out. The optimal configuration ($txpower_i = \{9, 9, 9, 3, 6, 3, 3, 6\}$, transmission power values in *dbm* for group *i*, see Figure 7.16 (b) for group locations) forms four super groups. One group, covering the left side of the network, is configured with the maximum transmission power. A second group, consisting of the top part of the network, is configured with the minimal power value. The bottom right part has also the minimal transmission power. The remaining nodes in the center and the right part of the network are configured with a medium power value.

Based on this configuration, the data is routed on two paths to the base station, see Figure 7.21 (c). The first path is located on the top edge of the network, while the second path starts in the center of the network and routes the data along the left side to the base station. Since only the data of the top three lines of nodes uses the top path, the workload on the different paths is very diverse causing more packet loss on the loaded path.

Distributed Groups Dynamic Table - Neighborhood

In Figure 7.22 (a) the average *Target Ratio* for the *Distributed Groups Dynamic Table - Neighborhood* is visualized. With an average value of 76.55%, the value is more than 20 percentage points higher than those of the equally configured nodes (55%). Since the individually configured network results in an average *Target Ratio* of 84.5% the differences between this model and an optimal configuration is below 10 percentage points. The required traffic is with 1.514mbps above the value of an optimal configuration (1.439mbps).

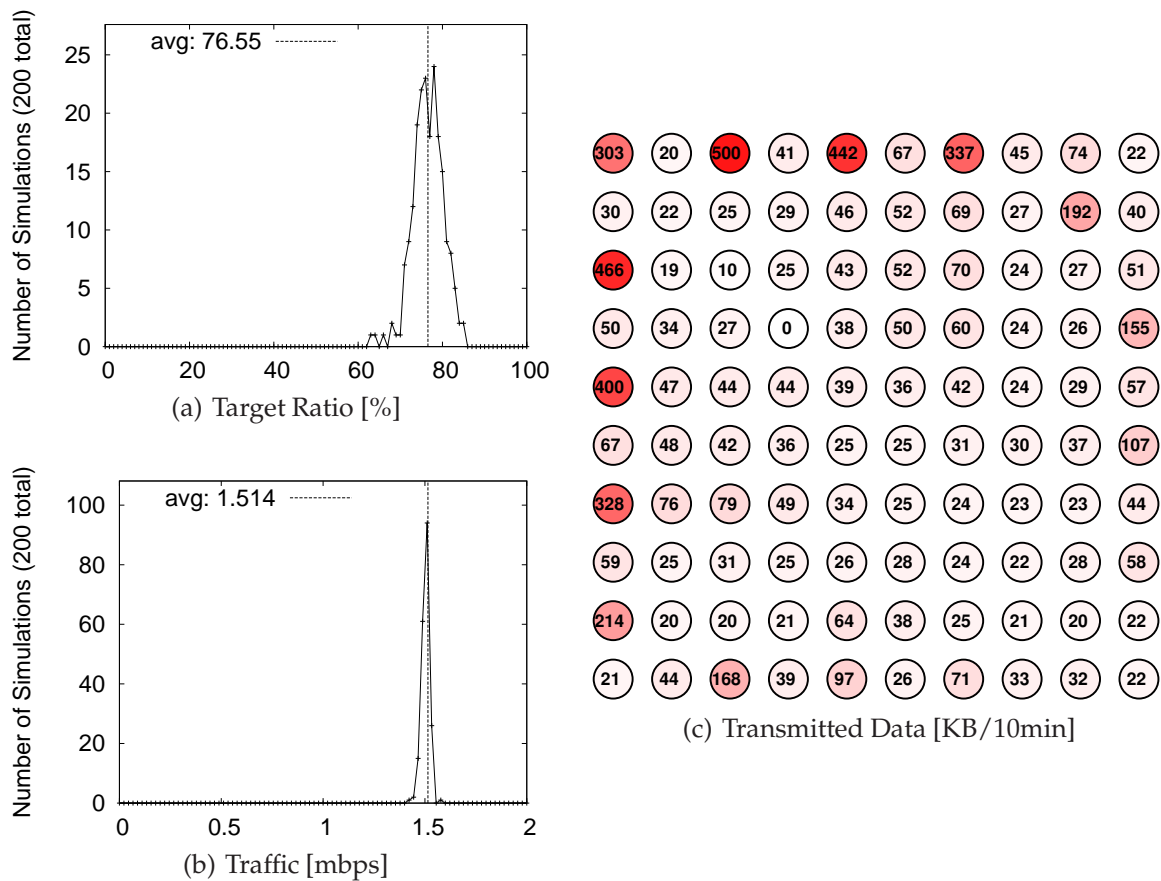


Figure 7.22: 10x10 nodes with model *DGDT-N* and optimal settings

The optimal configuration ($txpower_i = \{9, 3\}$, transmission power values in *dbm* for group *i*, see Figure 7.17 (a) for group locations) for the two formed groups configures the center of the network with a minimal transmission power. The edges of the network are configured with maximum power value. When comparing this configuration with the optimal individual configuration (Figure 7.10 (a)) only small differences are visible. The nodes on the bottom and the right edge should be configured with a low transmission power, and all nodes near the base station with a high power value.

A comparison of the data flow resulting from this model (Figure 7.22 (c)) with the data paths of the optimal individual configuration (Figure 7.10 (b)) yields in no noteworthy differences. Both configurations result in data flows routing the data to the edges and then to the base station. The data flow allows an explanation of the high required traffic. Since the *DGDT-N* model configures all nodes at the edges and not only the left and top part with a transmission power of $9dBm$, the nodes near the right and bottom edges overhear the total traffic generated by this part of the network.

Distributed Groups Dynamic Table - Traffic

When forming distributed groups based on the traffic distribution (*Distributed Groups Dynamic Table - Traffic*) an average *Target Ratio* of 60.07% is possible. The variation of the *Target Ratio* is similar to those of the *Location Based - Hexagon* model. Due to the low average value and the high variation, the benefit of this model against equally configured nodes is very limited.

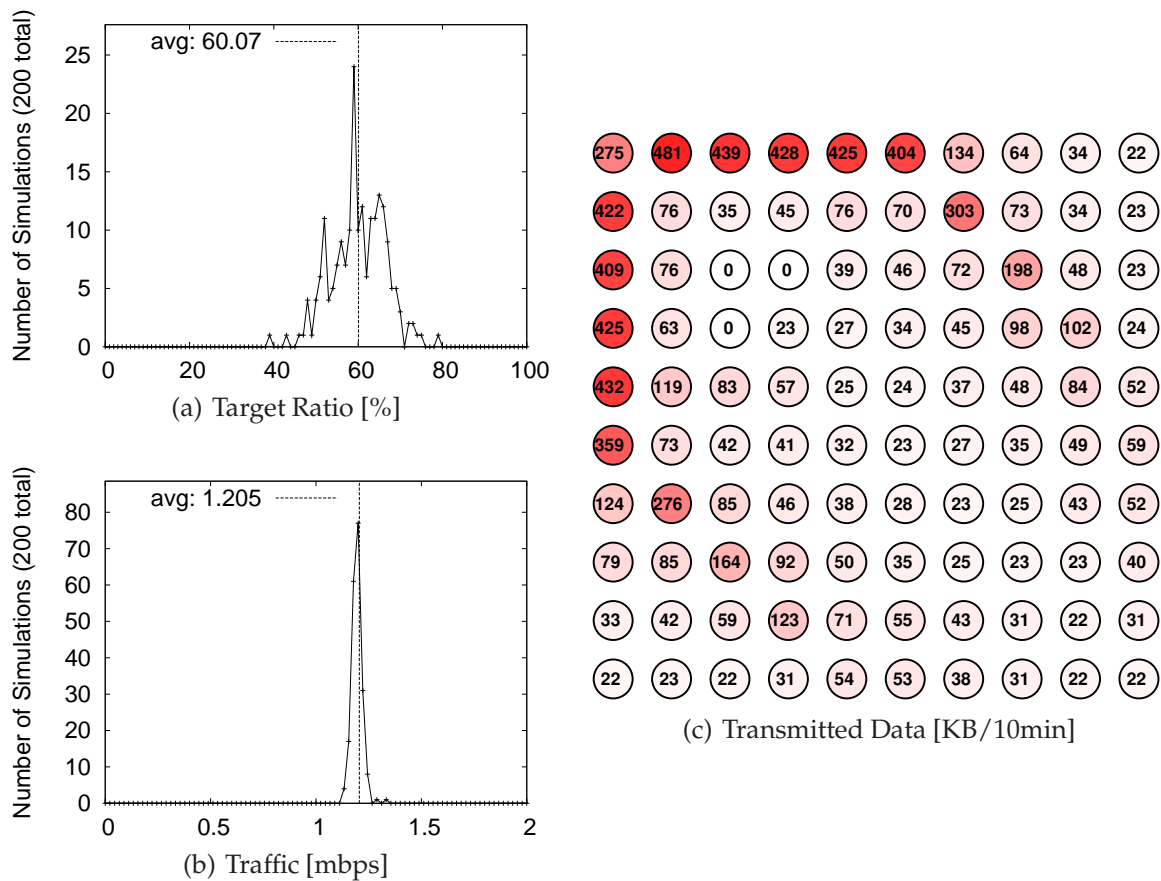


Figure 7.23: 10x10 nodes with model *DGDT-T* and optimal settings

The required traffic (1.205Mbps) is the lowest of all models. When examining the optimal configuration for the formed groups ($txpower_i = \{3, 3, 3, 6, 6, 6, 6, 3\}$, transmission power values in *dbm* for group *i*, see Figure 7.17 (b) for group location) the reason for the low traffic becomes visible. None of the nodes are configured with a transmission power of $9dBm$. The center and the nodes around the base station are configured with a power value of $3dBm$. The remaining nodes at the edges are configured with $6dBm$.

Due to the relative high values at the edges, the data is routed to the edges and from there to the base station. At the top right and bottom left corner the routing paths are diagonal. Keeping the location of the group with $3dBm$ in mind, it becomes obvious that the data is routed around this group.

7.3.5.2 Random Location

In the following, the results of the *Random Location* scenario are introduced.

Location Based - Circular

In Figure 7.24 (a) the average *Target Ratio*, resulting from optimal configured groups formed by the *Location Based - Circular* model, is visualized. With an average value of 83.65% the quality of this configuration is similar to the optimal individual configuration (84.97%). The variation of the results is about ± 8 which is also about the same level as an optimal individual configuration. The network requires about $0.866mbps$ traffic. Since the same configuration and individual configuration requires $0.879mbps$ and $0.843mbps$ respectively, this value is no outlier.

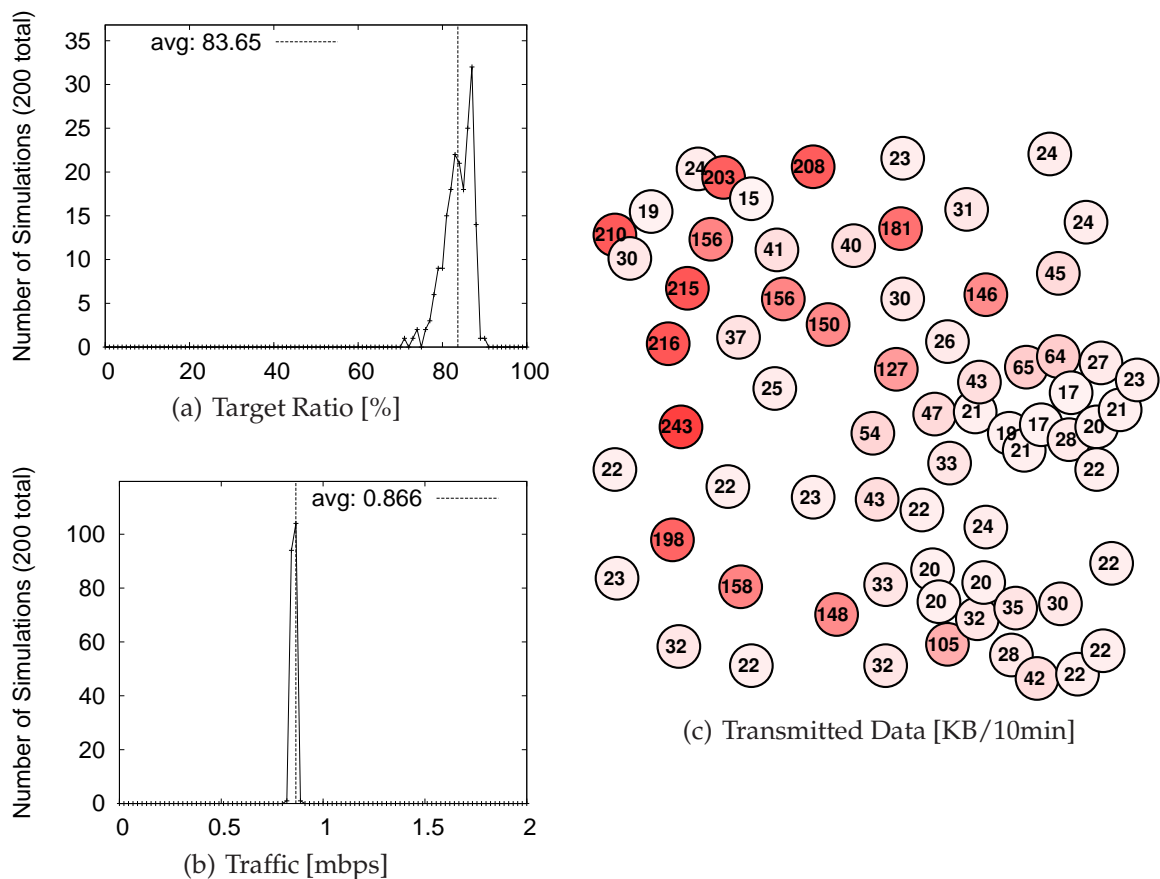


Figure 7.24: Randomly placed nodes with model *LB-C* and optimal settings

When analyzing the optimal configuration for the formed groups ($txpower_i = \{9, 6, 6, 6, 3, 6, 6, 3\}$, transmission power values in dbm for group i , see Figure 7.18 (a) for group location), it stands out that nearly the complete network is configured with a transmission power of $6dBm$. Only the base station with its neighborhood is configured with

9dBm. Nodes building Group 5 and 8, which are located in the center and right side of the network, are configured with a transmission power of 3dBm.

Figure 7.24 (c) visualizes the data flow generated by groups formed with this model. Using the optimal configuration for the formed groups, it is possible to distribute the traffic on three paths to the base station. With an individual configuration it is possible to use four data paths, which reduces the amount of data that has to be transmitted by most nodes. The benefit of the three paths formed by this model is that a lower number of paths reaches the base station and, therefore, reduces the number of collisions in the neighborhood of the base station.

Location Based - Hexagon

The average *Target Ratio* for groups created by the *Location Based - Hexagon* model is visualized in Figure 7.25 (a). When analyzing the distribution of the measurement results, two weaknesses attract attention. The first affects the low average of 74.08%, which is located ten percentage points below the results of other models. The second weakness is the relative high variation (± 12). With a required traffic of 0.855mbps, this model is about the same level as other configurations.

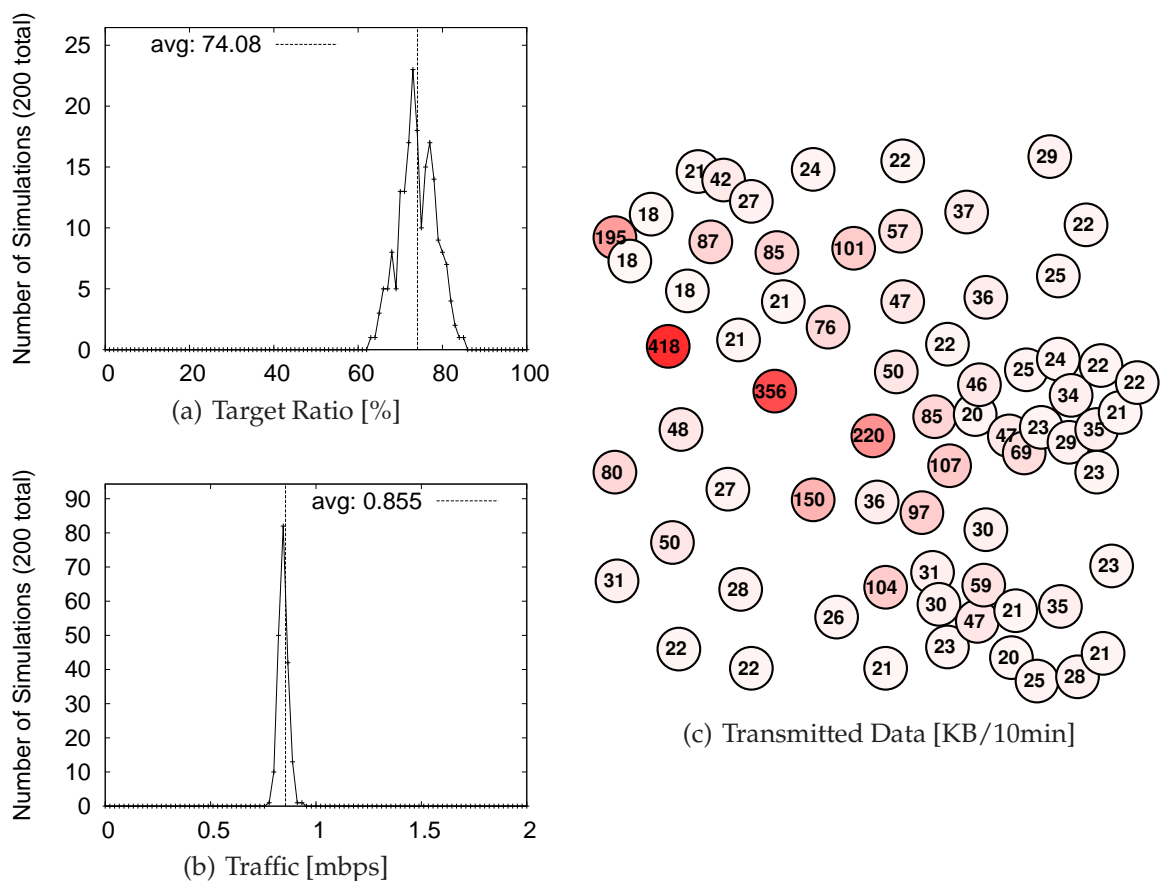


Figure 7.25: Randomly placed nodes with model *LB-H* and optimal settings

Like other models, *LB-H* configures the base station with a maximum transmission power. The optimal configuration ($txpower_i = \{9, 9, 9, 3, 6, 6, 3, 6\}$, transmission power values in

dbm for group i , see Figure 7.18 (b) for group location) sets up the left side of the network with a transmission power of $9dBm$. The center part and the lower and top right corner of the network are configured with the medium power value. The nodes in the top right part without the nodes in the corner are set up with a transmission power of $3dBm$.

Based on the introduced configuration, the data flow, shown in Figure 7.25 (c), is created. When analyzing the data paths, the weakness of this model becomes obvious. Most data is routed along one path. A second path at the top of the network transports only 20% of the lower path's traffic.

Distributed Groups Dynamic Table - Neighborhood

Figure 7.26 (a) shows the average *Target Ratio* when using the *Distributed Group Dynamic Table - Neighborhood* model for group formation. The average value is 82.9%, which is about two percentage points below the optimal individual configuration. The distribution of the values has a variation of ± 7 , which is comparable to those of the optimal setting. With a required traffic of $0.861mbps$, the amount of data created by this configuration is below the data needed by the equally configured nodes.

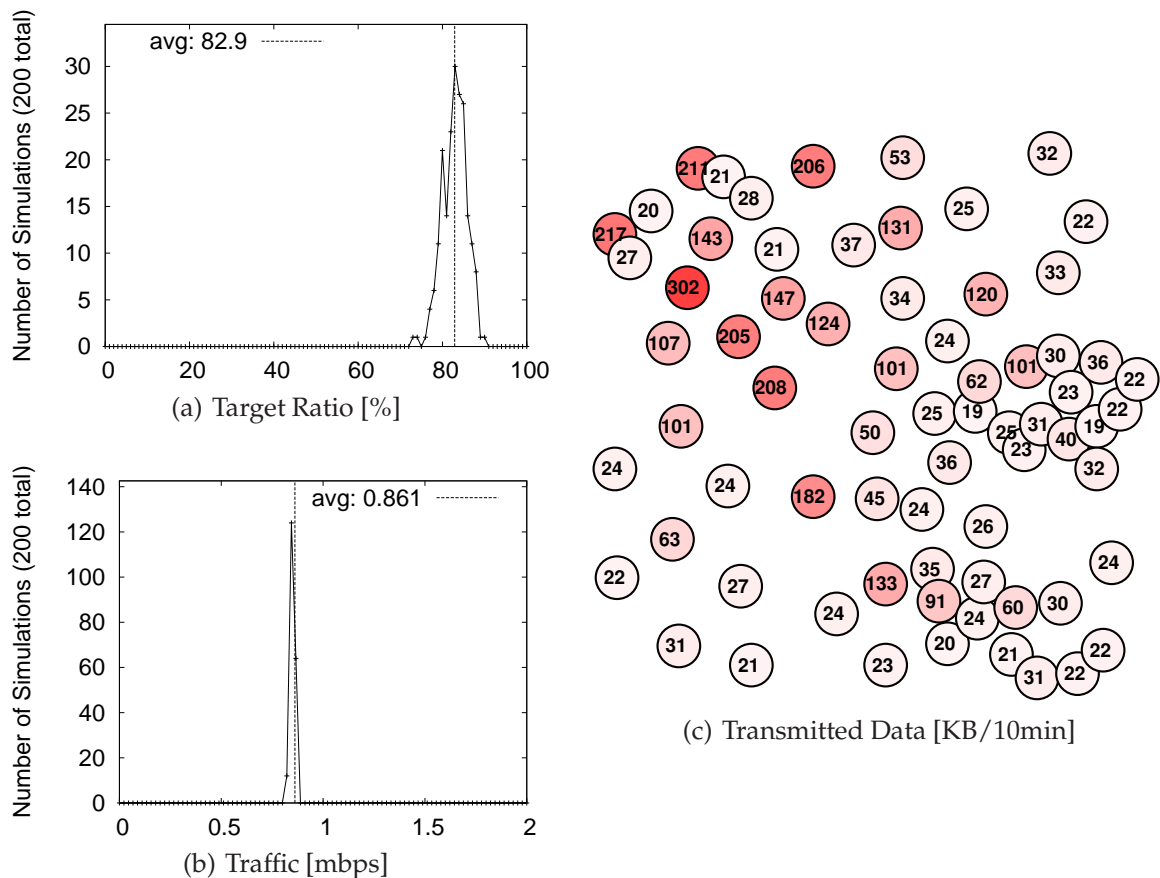


Figure 7.26: Randomly placed nodes with model *DGDT-N* and optimal settings

The optimal configuration for this model ($txpower_i = \{9, 6, 6, 6, 3, 3, 3, 6\}$, transmission power values in *dbm* for group i , see Figure 7.19 (a) for group location) configures the base station and the nodes at the bottom left and top right corner with $9dBm$. The center of the

two areas with high node density are configured with the minimal transmission power. All remaining nodes, located in areas with low node density, are set up with $6dBm$.

When analyzing the data flow, visualized in Figure 7.26 (c), the similarities with the data flow resulting from the optimal configuration attract attention. Both configurations create four data paths using the same nodes. The only difference is, that the most left path of the *DGDT-N* model starts on the bottom left side and, therefore, the second left path has to handle more data. Using the optimal configuration, this path also transports some data generated by the nodes at the bottom center of the network.

Distributed Groups Dynamic Table - Traffic

When using the *Distributed Groups Dynamic Table - Traffic* model, the average *Target Ratio* is 78.95%. The variation of the values is ± 7 . The weakness of this model in the *Random Location* scenario is the required traffic. While other configurations produce about $0.85mbps$ traffic this model requires $0.99mbps$.

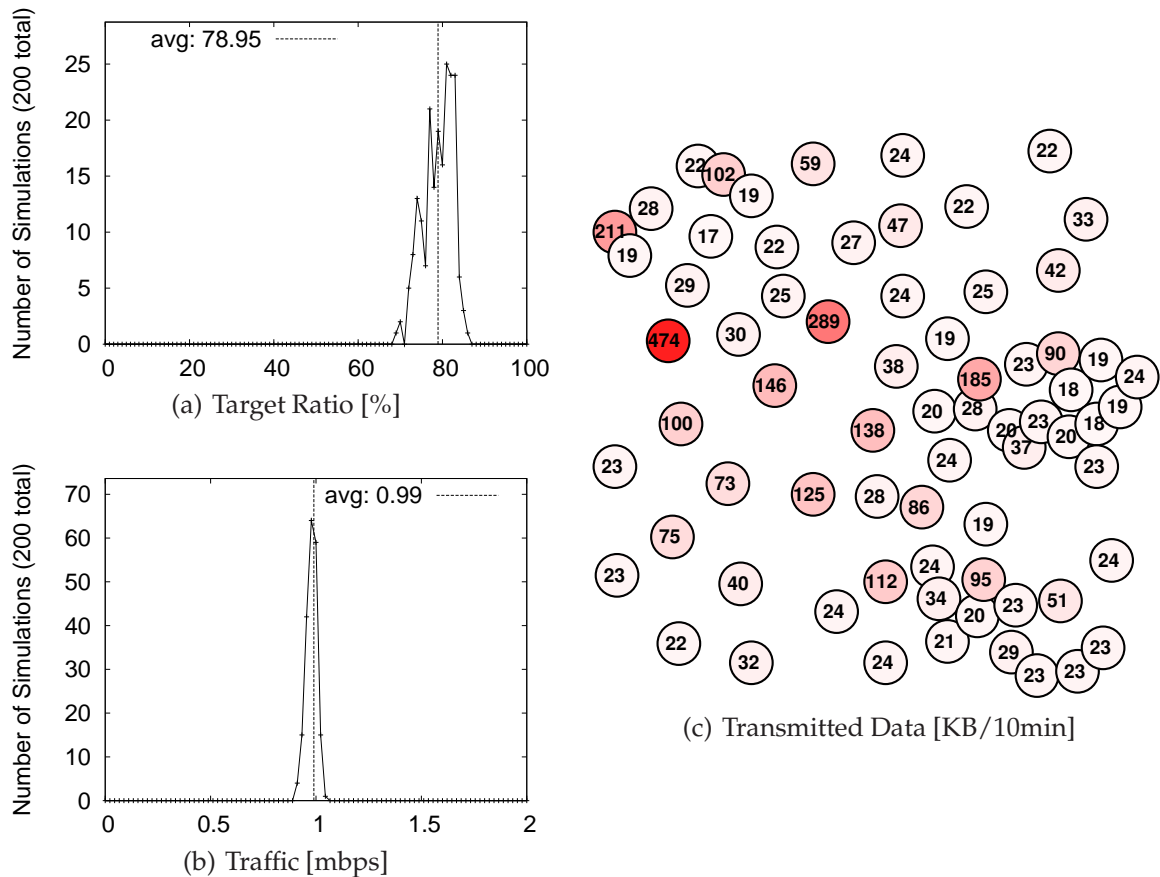


Figure 7.27: Randomly placed nodes with model *DGDT-T* and optimal settings

The analysis of the optimal configuration for this model ($txpower_i = \{9, 3, 3, 6, 3, 6, 6\}$, transmission power values in dbm for group i , see Figure 7.19 (b) for group location) shows that nodes with a transmission power of $9dBm$ are distributed in the network. The nodes with a transmission power of $3dBm$ are located at the edges. The remaining nodes in the center are

set up with a power value of $6dBm$. In contrast to the *Grid Pattern* scenario, here the nodes with a high traffic are configured with a high transmission power and nodes with low traffic with a low power value.

Since the routing metrics prefers nodes with a low hop distance to the base station for their parent node, most paths are using the nodes configured with $9dBm$. Figure 7.27 (c) shows the data flow resulting from the *DGDT-T* model. Due to the distributed nodes with a high sending range, the exact number of paths is difficult to count. The data is routed through the center using a network of paths. All paths are ending at a single node below the base station which forwards the data to the base station.

7.4 Discussion

7.4.1 Models

In the previous section, the different designed models were evaluated. During the evaluation the models are compared to the equally configured network on the one hand and an individually configured network on the other hand. The focus of this section is to sum up the pro and cons of the models.

The first result deduced from the evaluation is the huge potential for the group formation models to improve the performance of the network. In the *GridPattern* scenario an increment of the *Target Ratio* from about 55% to 85% is possible. The potential for optimizations in the *Random Location* scenario is about 15 percentage points (from 70% to 85%). When focusing on the *Random Location* scenario, the optimal individual configuration requires less traffic than the equally configured network.

A second important conclusion is the fact that the distributed approach, named *Most Equal Neighbor*, forms comparable groups as the *Distributed Groups Dynamic Table* models. Using the *Most Equal Neighbor* it is possible to form groups even in large scenarios where centralized approaches hardly scale.

Group Model	Grid Pattern		Random Location	
	Target Ratio	Traffic	Target Ratio	Traffic
Same Settings: $3dBm$	55.91%	1.026mbps	52.87%	0.765mbps
Same Settings: $6dBm$	54.38%	1.265mbps	69.66%	0.879mbps
Same Settings: $9dBm$	51.19%	1.783mbps	60.32%	1.085mbps
Optimal Groups:	84.51%	1.439mbps	84.97%	0.843mbps
Model LB-C:	72.88%	1.236mbps	83.65%	0.866mbps
Model LB-H:	60.03%	1.404mbps	74.08%	0.855mbps
Model DGDT-N:	76.55%	1.514mbps	82.90%	0.861mbps
Model DGDT-T:	60.07%	1.205mbps	78.95%	0.990mbps

Table 7.1: Model Evaluation Results

In Table 7.1 the results for different group models are listed. The top three lines contain the values for equally configured nodes followed by those for nodes with optimal individual settings. The target ratio and required traffic for the actual group formation models are listed at the bottom of the table. By analyzing the different measurements, the models can be divided into two groups. The results of the first group containing *LB-C* and *DGDT-N* models are much better than those of the *LB-H* and *DGDT-T* models.

A reconfiguration of transmission powers changes the routing paths, because some links are newly created and others are deleted. Since the traffic at nodes is a result of the routing paths, using the traffic as a foundation for group formation introduces a positive feedback. Based on the new traffic situation a new group formation is required. In the *Random Location* scenario this problem does not appear in a large manner and results in acceptable values. In contrast, the effect of the changing routing paths declassifies this model for usage in the scenario with regularly placed nodes, even if the required traffic is minimal.

When investigating the results of the *Location Based - Hexagon* model, only a minimal performance increase is visible. Due to the large potential of the group formation models and the performance of other models, the introduced overhead for building the groups is too high.

In Figure 7.28 and Figure 7.29 the quality of the remaining models is compared to the possible maximum and the quality without using groups. When considering only the reached *Target Ratio* the quality of the *LB-C* and *DGDT-N* models is almost equal. While the *DGDT-N* has little advantages in the regular formed scenario, the *LB-C* model has slight better results in the scenario with randomly distributed nodes. When taking also the required traffic into account, some differences between the models become visible. In the *Grid Pattern* scenario the traffic, required by the *DGDT-N* model, is much higher than that of the *LB-C*.

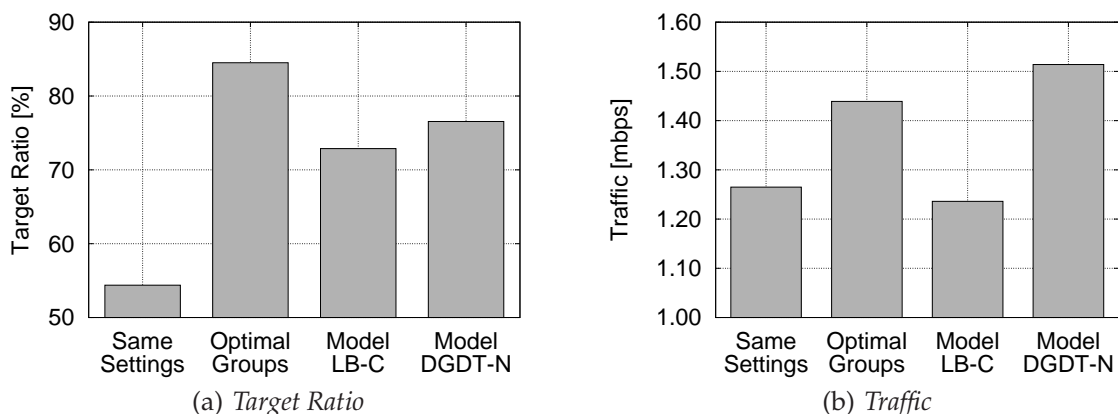


Figure 7.28: Model Evaluation Results: *Grid Pattern* scenario

The comparison of the *Target Ratio* with the optimal groups and the case where no groups are formed, shows that the usage of group formation models increases the performance of the network. In case of the *Grid Pattern* scenario and the *LB-C* model, see Figure 7.28 (b), the required traffic and, therefore, the needed energy decrease. The *Random Location* scenario shows that the performance increase of an individual configuration compared to those using groups of nodes is minimal, see Figure 7.29 (a).

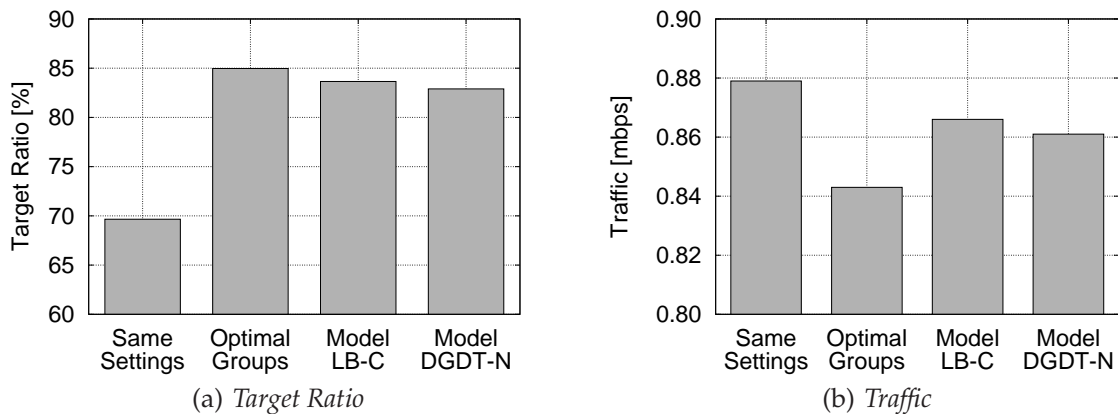


Figure 7.29: Model Evaluation Results: *Random Location* scenario

When rating the quality of the different models, the metrics focusing on the required traffic has to be used with caution. During the investigation of the optimal configuration for the groups the focus lays on maximizing the *Target Ratio*. The required traffic should be used to get an impression what overhead has to be paid to achieve this goal.

7.4.2 Generalization

After the discussing of the different models' quality, in the following general observations are introduced. Based on analysis of the different optimal configurations for the models, some general statements for good configurations and network partitions are possible.

In nearly all configurations the base station and nodes around it are set up with the maximum transmission power. The benefit of such a configuration can be explained by a view on the generated data paths. Since generally the data is routed on several paths to the base station, these paths have to transit the area around the base station. To minimize the number of collisions it is useful to minimize the number of hops in this area where the data paths are side by side and, because of this, influencing each other.

Another general observation is the goal of maximizing the number of data paths. Configurations with several independent paths result in a high *Target Ratio*. Independent means in this context that the paths are not side by side and, therefore, interfere each other. Such paths result in a high number of collisions and, therefore, in loss of data.

Since the amount of traffic on each routing path influences the number of collisions, the traffic on each path should be minimal. Based on the data flows resulting from configurations with a high *Target Ratio* a stronger statement can be deduced. The traffic should be equally distributed on all paths.

During the evaluation of the group formation models the search for optimal configurations is done manually. When integrating the group formation into *TinyCubus* an automatic configuration discovery is needed. Based on the optimal configurations for the differently grouped nodes, several general configuration patterns can be identified.

When grouping the network using the location of the nodes, it is possible to deduce an expedient setting using the distance to the base station. In both scenarios the nodes near the base station are configured with the maximum transmission power. With increasing distance the transmission power shrinks to the minimal value. An exception to this rule are the nodes at the edges. Since nodes at the opposite side of the base station are not used to forward any data, the configuration of these nodes is mostly irrelevant to the overall performance. The amount of packets generated by these nodes is very small and, therefore, a high transmission power does not cause many collisions and decreases the hop distance to the base station. Generally a distribution of the power values from a high over a medium to a low and again to a medium value results in a high *Target Ratio*.

The second configuration pattern is based on the density of nodes. Based on the evaluated scenarios, the following general conclusion can be deduced: Nodes with a high density should be configured with a low transmission power and vice versa. The effect of such a configuration is that the neighborhoods of the nodes are equalized.

Chapter 8

Conclusion

This chapter concludes this diploma thesis concerning group formation for adaptation purposes in wireless sensor networks. In the following section the whole topic is summarized and the most important results are recapitulated. Section 8.2 focuses on open questions and gives an overview on possible further research topics using this work as a foundation.

8.1 Summary

The topic of this work was to design group formation models creating clusters of nodes with equal properties. The guiding idea behind forming groups with equal properties is the assumption that nodes with equal properties should be configured with equal settings. Using these clusters it is possible to adapt the configuration of protocols efficiently, because the huge parameter space resulting from an individual configuration of every node ($O(c^n)$, where c denotes the number of different configurations per node and n the number of nodes in the network) is broken down to $O(c^{\frac{n}{s}})$, where s denotes the size of each group.

The first question addressed in this thesis was, therefore, which properties of a sensor node are suitable for usage. Based on this question, in Chapter 4 the different properties of a sensor network were discussed. This discussion covers network, node and application characteristics. During this disquisition it emerged that the structure of the network is a key feature influencing the performance of the system. The investigation for suitable properties finally has resulted in the identification of three promising properties: the location, the density and the amount of traffic at the nodes.

Based on the identified properties, in Part II different models were designed to form the actual groups. In general, all models are following the same basic guideline: Divide the network into partitions, so that all nodes belonging to the same partition have equal properties.

Following this basic guideline, several fundamentally different approaches for the actual group formation process have been designed. The design space covers everything from models forming distributed groups controlled by a centralized component over models supporting mobile nodes and ends with a distributed algorithm forming connected groups. To improve the performance of the developed models, several extensions to support scenarios with multiple base stations or obstacles were discussed. This part of the thesis was closed by a theoretical discussion of the developed models. During the discussion it turned out

that the location based models are fulfilling most requirements for optimal groups. A second and expected conclusion was the fact that centralized approaches require much more messages to be sent than distributed or hybrid approaches. Based on the discussion it can be stated that the models forming distributed groups and the location based models are the best candidates to form groups for adaptation purposes in wireless sensor networks.

Since sensor networks are huge distributed systems with dozens of nodes, the theoretical discussion could not cover the dynamic aspects of the network and, therefore, the quality of the models was evaluated by simulating the network with the grouped and adapted nodes (Part III). The protocol whose behavior was adapted, was a routing protocol. The adaptation was performed by tuning the transmission range of the nodes. To measure the quality of the models, different scenarios covering typical network topologies have been designed. The measurement of the quality itself is providing several metrics, like the required traffic or the percentage of data reaching the base station. To allow a quantification of the benefit introduced by the models, the quality of the approaches was compared to scenarios without group formation and individually configured nodes.

During the evaluation it pointed out that forming nodes based on the location and the neighborhood of nodes gives best results. In the best case the percentage of delivered data increases from 70% (no groups) to 83.65%. An individual configuration of the nodes is able to outreach this value only by 1.32 percentage points. The models forming distributed groups using the neighborhood metrics (*DGDT-N*) and the location based model arranging the nodes in circles around the base station (*LB-C*) provided best results in all scenarios. Since the former model requires more traffic in the scenario with regularly placed nodes, the location based model wins by a narrow margin. Based on the results gained during the evaluation, it can be stated that introducing a group formation component allows to reduce the huge parameter space of individually configured nodes without losing the potential of adapting the nodes.

8.2 Limitations and Future Work

The focus of this thesis was the development of group formation models for adaptation purposes. The basic idea behind the models is to group nodes which have equal properties. Since the properties are depending on the algorithm class to be adapted and the huge size of possible algorithm categories, one algorithm class has to be exemplarily chosen. In case of this work, the selection was the class of routing algorithms, because routing is required in nearly every sensor network. Since most models are designed to work with an arbitrary metrics for identification of nodes with equal properties, the models are suitable for other algorithm classes, too.

An open question is rather the issue, which properties should be used to determine the group membership. In case of the class of routing algorithms these properties have been identified during this thesis. In further research activities the properties for other classes have to be identified and evaluated, too.

Another important question is how to identify the optimal settings for a group. During the evaluation of this work, this job was provided by a human controlled, generic algorithm. When integrating the group formation component into *TinyCubus* an automatic process to

determine the configuration is required. During the discussion of the evaluation results, some general statements about good configurations were made.

During the evaluation of the models it became obvious that the performance increase of the algorithm is issued by changing the routing paths. Since the formation of routing paths depends on the actual implementation of the routing algorithm or rather the used routing metrics, the statements for optimal configuration are in particular valid for the used routing algorithm. In further work the observed rules have to be checked by using other algorithms.

Since this thesis is intended as a case study, the group formation models are implemented prototypically. An integration of the model into the *TinyCubus* framework requires a port of the code to *NesC* and the *TinyCubus* environment.

List of Tables

4.1	Common sensor node hardware platforms	18
4.2	Battery based power supply	19
4.3	Comparison: computing data vs transmitting data	19
5.1	Required messages for electing a cluster head	33
5.2	Fulfilled Neighborhood Value Requirements	35
6.1	Group Model Discussion - Supported Features	60
6.2	Group Model Discussion - Message Overhead Key	61
6.3	Group Model Discussion - Message Overhead	61
7.1	Model Evaluation Results	97
C.1	Position of the randomly placed nodes	XI

List of Figures

2.1	Grouping Component	5
2.2	Sample application based on TinyOS	6
2.3	Components of TinyCubus	7
2.4	Architecture of TinyCubus	8
4.1	Sensor nodes	16
4.2	Structured Node Deployment: Monitoring crop growing	21
4.3	Random Node Deployment: Monitoring forest fire	21
4.4	Moving Groups	22
4.5	Connection time between moving nodes	23
5.1	Radio signal propagation	29
5.2	Weighted, undirected graph	31
5.3	Separated graph	32
5.4	Cluster Head Election	32
5.5	Neighborhood Value: Sample Network	34
5.6	Possible <i>Neighborhood Value</i> metrics (1)	34
5.7	Possible <i>Neighborhood Value</i> metrics (2)	35
5.8	Group Borders: Small Changes	37
5.9	Optimal group borders	37
6.1	Group Formation Algorithm Classification	39
6.2	Most Equal Neighbor: Location	42
6.3	Most Equal Neighbor: Neighborhood	42
6.4	Most Equal Neighbor: Traffic	43
6.5	Distributed Groups Static Table - Neighborhood	45
6.6	Mobile Groups: Moving Nodes	46
6.7	Mobile Groups: Metrics using Connection Time	46
6.8	Distributed Groups Dynamic Table - Neighborhood	49
6.9	Distributed Groups Dynamic Table - Traffic	49
6.10	Generic Location Based Model	51
6.11	Location Based Model: <i>Hexagon Pattern</i>	51
6.12	Location Based Model: <i>Circular Pattern</i>	52
6.13	Weighted Links - Graph Partitioning: Neighborhood	53
6.14	Weighted Links - Graph Partitioning: Traffic	55
6.15	Environment Based Model	57
6.16	Different Node Types Model	58

7.1	Scenario Evaluation Components	65
7.2	<i>Grid Pattern</i> scenario: Node deployment and transmission ranges	71
7.3	<i>Random Location</i> scenario: Node deployment and transmission ranges	73
7.4	10x10 nodes with same configuration (TXPower 3dBm)	75
7.5	10x10 nodes with same configuration (TXPower 6dBm)	75
7.6	10x10 nodes with same configuration (TXPower 9dBm)	76
7.7	Random placed nodes with same configuration (TXPower 3dBm)	76
7.8	Random placed nodes with same configuration (TXPower 6dBm)	77
7.9	Random placed nodes with same configuration (TXPower 9dBm)	77
7.10	Optimal configuration for 10x10 nodes	78
7.11	10x10 nodes with individual configuration	79
7.12	Optimal configuration for randomly placed nodes	80
7.13	Randomly placed nodes with individual configuration	81
7.14	Group formation metrics for 10x10 Nodes with TXPower 6dBm	82
7.15	Group formation metrics for randomly placed nodes with TXPower 6dBm	83
7.16	Grouped 10x10 nodes with model: LB-C, LB-H, MEN-N, MEN-T	84
7.17	Grouped 10x10 nodes with model: DGDT-N, DGDT-T, DGST-N	85
7.18	Grouped random located nodes with model: LB-C, LB-H, MEN-N, MEN-T	86
7.19	Grouped random located nodes with model: DGDT-N, DGDT-T, DGST-N	87
7.20	10x10 nodes with model <i>LB-C</i> and optimal settings	89
7.21	10x10 nodes with model <i>LB-H</i> and optimal settings	90
7.22	10x10 nodes with model <i>DGDT-N</i> and optimal settings	91
7.23	10x10 nodes with model <i>DGDT-T</i> and optimal settings	92
7.24	Randomly placed nodes with model <i>LB-C</i> and optimal settings	93
7.25	Randomly placed nodes with model <i>LB-H</i> and optimal settings	94
7.26	Randomly placed nodes with model <i>DGDT-N</i> and optimal settings	95
7.27	Randomly placed nodes with model <i>DGDT-T</i> and optimal settings	96
7.28	Model Evaluation Results: <i>Grid Pattern</i> scenario	98
7.29	Model Evaluation Results: <i>Random Location</i> scenario	99
A.1	Transmitted data: equally configured nodes [KB/10min] (1)	VI
A.2	Transmitted data: equally configured nodes [KB/10min] (2)	VII
A.3	Transmitted data: equally configured nodes [KB/10min] (3)	VII

List of Algorithms

5.1	Graph Partitioning	31
7.1	TrafficGenerator Configuration	66
7.2	Spanning Tree Construction	67
7.3	Medium Configuration	69
B.1	Scenario file body	VIII
B.2	Template for a node	IX
B.3	Protocol stack used of traffic generating nodes	IX
B.4	Protocol stack used of the base station	X

Part IV

Appendix

Appendix A

No Groups - Equal Settings

In the following figures the required traffic for equally configured nodes is visualized. The traffic is visualized for both scenarios, *Grid Pattern* and *Random Location* as well as for all three transmission powers. The corresponding *Target Ratio* is visualized and discussed in Section 7.3.1.

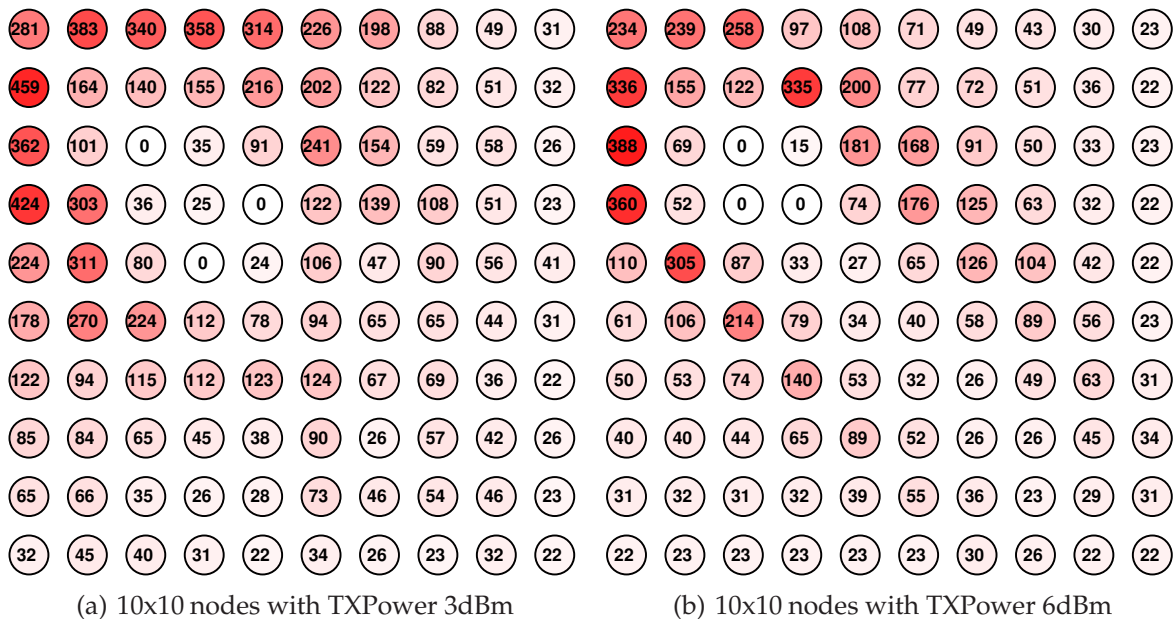


Figure A.1: Transmitted data: equally configured nodes [KB/10min] (1)

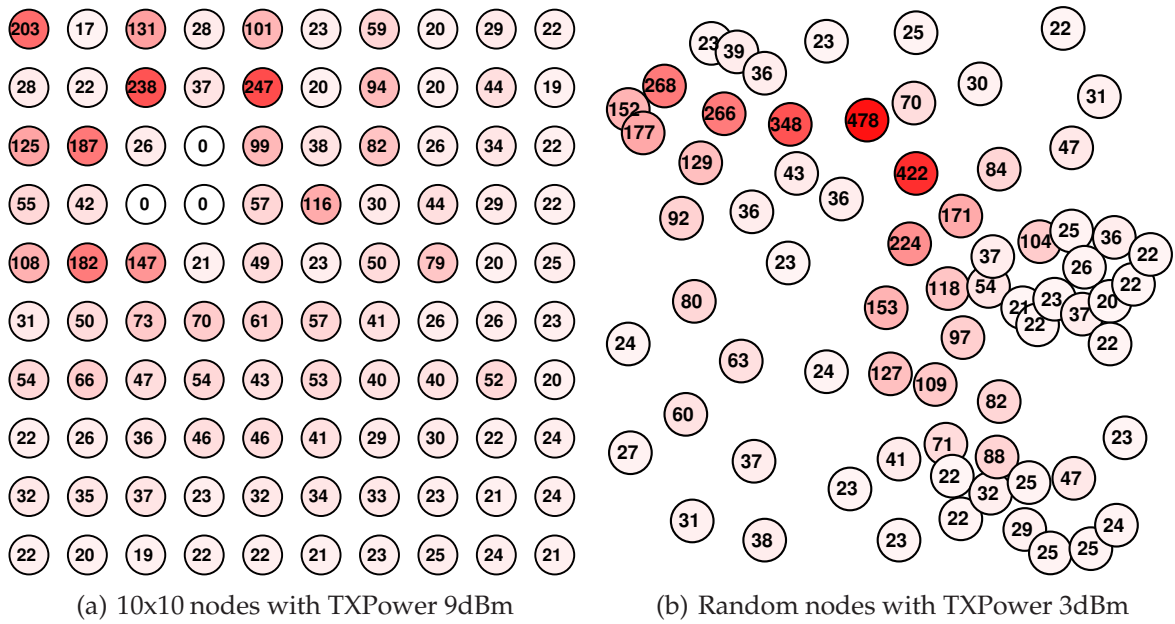


Figure A.2: Transmitted data: equally configured nodes [KB/10min] (2)

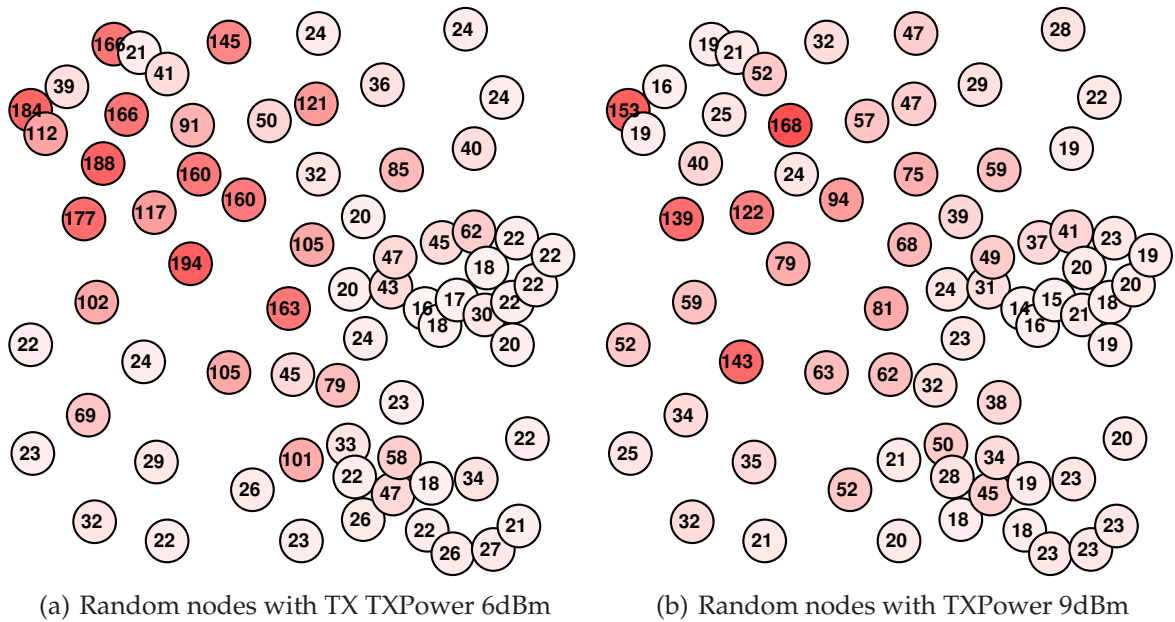


Figure A.3: Transmitted data: equally configured nodes [KB/10min] (3)

Appendix B

CUBUS scenario files

```

1 <engine value="SimulationEngine">
2   <plugins value="ObjectList">
3     <item0 value="/TreeRoutingPrinter">
4       <zero_stat_time value="600000"/>
5       <time value="1199999"/>
6     </item0>
7     <item1 value="/TinyCubus">
8       <initPhase value="300000"/>
9       <adaptNodesTime value="300000" />
10      <testTime value="600000" />
11      <number_adaptation_rounds value="1"/>
12      <adaptation_model value="RANDOM"/>
13    </item1>
14    <item2 value="/GroupFormation">
15      <model value="LB_C" />
16      <time value="299998" />
17    </item2>
18  </plugins>
19  <simulationFrame value="/Frame">
20    <mediums value="ObjectList">
21      <item0 value="/Wireless">
22        <transmissionRate value="154" />
23      </item0>
24    </mediums>
25    <nodes value="ObjectList">
26      <!-- node definitions -->
27    </nodes>
28  </simulationFrame>
29  <maxSimulationTime value="12m" />
30 </engine>

```

Algorithm B.1: Scenario file body

```

1 <item0 value="/Node">
2   <pos value="Position">
3     <x value="X" />
4     <y value="Y" />
5   </pos>
6   <stack value="PROTOCOLSTACK-TEMPLATE.XML" />
7 </item0>

```

Algorithm B.2: Template for a node

```

1 <stack value="ProtocolStack">
2   <protocols value="ObjectList">
3     <item0 value="/IEEE80211" >
4       <maxRetransmits value="3"/>
5     </item0>
6     <item1 value="/TreeRouting">
7       <beacon_send_interval value="3000"/>
8       <max_ttl value="10"/>
9     </item1>
10    <item2 value="/TrafficGenerator">
11      <minPackageSize value="30" />
12      <maxPackageSize value="30" />
13      <byteTraffic value="3" />
14      <includeUniqueIdentifier value="1" />
15    </item2>
16  </protocols>
17  <mediumLinks value="ObjectList">
18    <item0 value="MediumLink">
19      <medium value="../../../../mediums/item0" />
20      <algorithm value="../../protocols/item0" />
21      <maxRange value="100"/>
22      <antenna value="/Antenna">
23        <sendingPower value="6"/>
24      </antenna>
25    </item0>
26  </mediumLinks>
27  <protocolLinks value="ObjectList">
28    <item0 value="ProtocolLink">
29      <algorithm1 value="../../protocols/item1" />
30      <algorithm2 value="../../protocols/item0" />
31    </item0>
32    <item1 value="ProtocolLink">
33      <algorithm1 value="../../protocols/item2" />
34      <algorithm2 value="../../protocols/item1" />
35    </item1>
36  </protocolLinks>
37 </stack>

```

Algorithm B.3: Protocol stack used of traffic generating nodes

```

1 <stack value="ProtocolStack">
2   <protocols value="ObjectList">
3     <item0 value="/IEEE80211" >
4       <maxRetransmits value="3"/>
5     </item0>
6     <item1 value="/TreeRouting">
7       <beacon_send_interval value="3000"/>
8       <max_ttl value="10"/>
9     </item1>
10    <item2 value="/TrafficLogger">
11      <uniqueData value="1" />
12    </item2>
13  </protocols>
14  <mediumLinks value="ObjectList">
15    <item0 value="MediumLink">
16      <medium value="../../../../../../mediums/item0" />
17      <algorithm value="../../protocols/item0" />
18      <maxRange value="100"/>
19      <antenna value="/Antenna">
20        <sendingPower value="6"/>
21      </antenna>
22    </item0>
23  </mediumLinks>
24  <protocolLinks value="ObjectList">
25    <item0 value="ProtocolLink">
26      <algorithm1 value="../../protocols/item1" />
27      <algorithm2 value="../../protocols/item0" />
28    </item0>
29    <item1 value="ProtocolLink">
30      <algorithm1 value="../../protocols/item2" />
31      <algorithm2 value="../../protocols/item1" />
32    </item1>
33  </protocolLinks>
34 </stack>

```

Algorithm B.4: Protocol stack used of the base station

Appendix C

Node positions in the *Random Location* scenario

Node ID	Position		Node ID	Position		Node ID	Position	
	X	Y		X	Y		X	Y
0	0	0	1	0	150	2	1	201
3	100	82	4	104	218	5	112	45
6	121	133	7	123	164	8	127	204
9	127	242	10	132	103	11	134	37
12	135	4	13	135	70	14	144	169
15	149	197	16	150	124	17	152	212
18	156	232	19	156	90	20	157	147
21	165	28	22	169	123	23	17	29
24	170	220	25	171	109	26	173	203
27	174	177	28	174	68	29	185	133
30	186	237	31	188	215	32	192	141
33	193	102	34	198	248	35	200	129
36	204	2	37	208	97	38	209	213
39	213	136	40	214	114	41	217	246
42	208	58	43	221	34	44	226	130
45	226	150	46	228	100	47	229	235
48	233	194	49	237	122	50	245	108
51	25	91	52	27	183	53	30	233
54	31	130	55	34	65	56	39	9
57	53	158	58	45	42	59	51	13
60	58	88	61	59	205	62	64	23
63	64	242	64	7	51	65	75	112
66	76	47	67	79	70	68	93	163
69	93	8						

Table C.1: Position of the randomly placed nodes

Appendix D

Bibliography

- [ABS03] S. Ganeriwal, A. Boulis, and M. B. Srivastava. Aggregation in sensor networks: An energy-accuracy trade-off. In *First IEEE International Workshop on Sensor Network Protocols and Applications (SNPA 2003)*, Anchorage, AK, USA, May 11 2003.
- [ASSC02] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks (Amsterdam, Netherlands: 1999)*, 38(4):393–422, 2002.
- [BCSW98] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (dream). In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 76–84, New York, NY, USA, 1998. ACM Press.
- [Bri07] Sustainable Bridges. Internet <http://www.sustainablebridges.net>, 2007.
- [Cro06a] Crossbow. Mica2 datasheet. December 2006.
- [Cro06b] Crossbow. Telos revb datasheet. December 2006.
- [Cro06c] Crossbow. Telosb datasheet. December 2006.
- [CUB07] CUBUS-Team, Computer Science, University Stuttgart. CUBUS Simulator. Internet <http://www.ipvs.uni-stuttgart.de/abteilungen/vs/abteilung/mitarbeiter/lehre/lehrveranstaltungen/studienprojekte/CUBUS>, March 2007.
- [DKN03] Koustuv Dasgupta, Konstantinos Kalpakis, and Parag Namjoshi. An efficient clustering-based heuristic for data gathering and aggregation in sensor networks. In *Proceedings IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, LA, 2003.
- [Els97] Ulrich Elsner. Graph partitioning - a survey. December 1997.
- [GM82] Hector Garcia-Molina. Elections in a Distributed Computer System. *IEEE Transactions on Computers C-31(2):48-59*, 1982.
- [HMCP04] W. Heinzelman, A. Murphy, H. Carvalho, and M. Perillo. Middleware to support sensor network applications. *IEEE Network Mag.*, 18(1):6–14, 2004.

- [HSW⁺00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System Architecture Directions for Networked Sensors. in ASPLOS. In *Information Technology*, November 2000.
- [JMC⁺01] P. Jacquet, P. Mühlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. Optimized link state routing protocol for ad hoc networks. In *Proceedings of the 5th IEEE Multi Topic Conference (INMIC 2001)*, 2001.
- [LM03] T. Liu and M. Martonosi. Impala: A middleware system for managing autonomous, parallel sensor systems. *ACM SIGPLAN Symp. Principles and Practice of Parallel Programming*, June 2003.
- [MDGS06] Gaurav Mathur, Peter Desnoyers, Deepak Ganesan, and Prashant Shenoy. Ultra-low power data storage for sensor networks. In *IPSN '06: Proceedings of the fifth international conference on Information processing in sensor networks*, pages 374–381, New York, NY, USA, 2006. ACM Press.
- [MHD⁺03] Peter Morsink, Redouane Hallouzi, Ismail Dagli, Christian Cseh, Lorenz Schäfers, Martin Nelisse, and Dik de Bruin. CARTALK 2000: Development of a Cooperative ADAS based Vehicle-to-Vehicle Communication. In *10th World Congress and Exhibition on Intelligent Transport Systems and Services*, November 2003.
- [MLM⁺05] Pedro José Marrón, Andreas Lachenmann, Daniel Minder, Jörg Hähner, Robert Sauter, and Kurt Rothermel. TinyCubus: A flexible and adaptive framework for sensor networks. In *Proceedings of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)*, pages 278–289, January 2005.
- [MMLR05] Daniel Minder, Pedro José Marrón, Andreas Lachenmann, and Kurt Rothermel. Experimental construction of a meeting model for smart office environments. In *Proceedings of the First Workshop on Real-World Wireless Sensor Networks (REALWSN 2005)*, SICS Technical Report T2005:09, June 2005.
- [MPS⁺02] Alan Mainwaring, Joseph Polastre, Robert Szewczyk, David Culler, and John Anderson. Wireless sensor networks for habitat monitoring. In *First ACM Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, USA, Sep. 2002.
- [MR04] V. Mhatre and C. Rosenberg. Homogeneous vs Heterogeneous Clustered Sensor Networks: A Comparative Study. In *IEEE International Conference on Communications*, pages 3646–3651 Vol. 6, 2004.
- [MSKG05] Pedro José Marrón, Olga Saukh, Markus Krüger, and Christian Große. Sensor network issues in the Sustainable Bridges project. In *European Projects Session of the Second European Workshop on Wireless Sensor Networks (EWSN 2005)*, January 2005.
- [Nea06] Jonas Neander. Using existing infrastructure as support for wireless sensor networks. Licentiate thesis, June 2006.
- [Per97] C. Perkins. Ad-hoc on-demand distance vector routing. *MILCOM '97 panel on Ad Hoc Networks*, November 1997.

- [PIP⁺03] N. Patwari, A. III, M. Perkins, N. Correal, and R. O’Dea. Relative location estimation in wireless sensor networks. *IEEE Transactions on Signal Processing*, August 2003.
- [PSC05] Joseph Polastre, Robert Szewczyk, and David Culler. Telos: Enabling ultra-low power wireless research. *The Fourth International Conference on Information Processing in Sensor Networks: Special track on Platform Tools and Design Methods for Network Embedded Sensors (IPSN/SPOTS)*, April 2005.
- [RKM02] Kay Römer, Oliver Kasten, and Friedemann Mattern. Middleware challenges for wireless sensor networks. In *Mobile Computing and Communications Review*, Vol. 6, Nr. 2, 2002.
- [RZL06] Rodrigo Roman, Jianying Zhou, and Javier Lopez. Applying Intrusion Detection Systems to Wireless Sensor Networks. In *IEEE CCNC 2006/IEEE Communications Society*, 2006.
- [SWR98] Suresh Singh, Mike Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 181–190, 1998.
- [TEC06] TECO. upart 014x ilmt preliminary datasheet. December 2006.
- [TS05] Kirsten Terfloth and Jochen Schiller. Driving forces behind middleware concepts for wireless sensor networks. In *Proceedings of the REALWSN Workshop*, 2005.
- [Xu03] Guochang Xu. *GPS. Theory, Algorithms and Applications*, ISBN 3540678123. Juni 2003.
- [YF04] O. Younis and S. Fahmy. Distributed clustering in ad-hoc sensor networks: A hybrid, energy-efficient approach. In *Proceedings of the IEEE Conference on Computer Communications (INFOCOM)*, Hongkong, 2004.
- [YG03] Yong Yao and Johannes Gehrke. Query processing for sensor networks. In *First Biennial Conference on Innovative Data Systems Research (CIDR 2003)*, 2003.
- [ZGE02] Y. J. Zhao, R. Govindan, and D. Estrin. Residual energy scan for monitoring sensor networks. In *IEEE Wireless Communications and Networking Conference (WCNC’02)*, March 2002.
- [ZGE03] Y. J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. *Technical Report 02-773, USC*, September 2003.

Appendix E

Statement

I ensure that I have created this document on my own and only used those external sources listed in the bibliography.

Stuttgart, _____

Andreas Grau