On group formation for self-adaptation in pervasive systems (Invited Paper)

Daniel Minder
Universität Bonn
Institut für Informatik IV
Römerstr. 164
53117 Bonn, Germany
minder@cs.uni-bonn.de

Andreas Grau
Universität Stuttgart
Institut für Parallele und Verteilte Systeme
Universitätsstr. 38
70569 Stuttgart, Germany
andreas.grau@ipvs.uni-stuttgart.de

Pedro José Marrón
Universität Bonn
Institut für Informatik IV
Römerstr. 164
53117 Bonn, Germany
pjmarron@cs.uni-bonn.de

ABSTRACT

Adaptation in Pervasive Computing normally focuses on services or on application behaviour, but the consideration of lower level algorithms in this process can lead to significant performance increase. To perform adaptation of algorithms the concept of context normally used in Pervasive Computing has to be extended. Based on the same context, group formation models are established to group devices with similar relevant context to optimise the adaptation process. In this paper, such group formation models are developed and evaluated. We also draw general conclusions for adaptation.

1. INTRODUCTION

Over the last years the vision of disappearing computers, which seamlessly interact with the environment and the user, has become more and more of a reality. Some common applications developed as part of this vision include smart environments, navigation and tourist guide systems, virtual information tags, etc. These applications mainly consider the selection of information and services, the presentation of such information and autonomic actions based on context.

At the same time, Wireless Sensor Networks have emerged as a way to gather data from the real-world, for example, the behaviour of animals, observation of environmental phenomena, monitoring of vital parameters or the tracking of goods.

As these examples show, both research topics are merging, not only because Pervasive Computing uses Wireless Sensor Networks as information providers, but also because context management and action functionality is moving into the sensor network. The Embedded Wisents project has, therefore, introduced the notion of Cooperating Objects that comprises both worlds [8].

The adaptation of application behaviour, which is an important aspect of Pervasive Computing applications, appears again in Wireless Sensor Networks. Here, algorithms and their parameterisations are subject to adaptation to improve mainly the performance or the power consumption of the network. We argue that Pervasive Computing application can also benefit from such adaptive behaviour at a lower level.

Pervasive Computing applications are often characterised by periods of static behaviour that are interrupted by dynamic states in which the network changes. For example, a smart room can be considered as static as long as no device leaves or enters the room. In the EMMA project [4], a car itself is a static Cooperating Object that is only changed during maintenance. After such a network change, the system has to reconfigure. In a service-centric application, services might no longer be available and have to be replaced by others. But also algorithms have to be adapted so that their behaviour becomes close to optimal again.

To allow for the adaptation of algorithms, "context" must not only be considered as information directly relevant for the user or affecting the behaviour of the visible application, but it has to be extended to characteristics of the environment that influence only the interior of an algorithm, which is usually not visible to the superordinate application.

TinyCubus [7], started while the authors were at the University of Stuttgart, provides such a framework targeted for Wireless Sensor Networks and Cooperating Objects that allows for the adaptation of algorithms. Based on network conditions, user requirements and algorithm characteristics, appropriate algorithms or its parameterisations are selected and installed. In [9], we have shown that adaptation can be performed on each node individually, on the network as a whole or on groups of nodes. The latter turned out to give good results while keeping a low overhead.

This paper analyses more deeply how such groups that are suitable for adaptation can be formed in the network. After the discussion of related work in Chapter 2, the necessary context information for adaptation is shown in Chapter 3. Chapter 4 presents possible group formation models and assesses them. The evaluation in Chapter 5 first explains the experimental setup, then shows the optimal configuration on a single node basis and the performance of several of group formations and finally discusses the results. The paper is concluded with a summary and outlook in Chapter 6.

Published in Proceedings of the 1st international conference on Autonomic computing and communication systems (Autonomics'07), Rome, Italy, 2007
© ICST, Brussels, Belgium, Belgium, 2007
http://portal.acm.org/citation.cfm?doid=1365562.1365584

2. RELATED WORK

The topic of adaptive systems is covered by a variety of papers. In this section, we give an overview of relevant projects in Wireless Sensor Networks and in Pervasive Computing.

As discussed before, adaptive middleware for Wireless Sensor Networks focuses on algorithms and parameters. Impala [6] is based on a finite state machine. It regularly queries the parameters in the condition rules from the application or the system, evaluates the rules and switches to another application if a rule is satisfied. Our approach using TinyCubus is more flexible than the finite state machine and allows for the exchange or parameterisation of a single algorithm and not on the coarse application level. Moreover, coordination between nodes like in our group approach is not done in Impala.

The MiLAN [5] middleware allows applications to specify their quality needs and the various sensors to announce the quality of their sensor data. MiLAN determines the sensors satisfying the application's quality requirements and selects among them the data provides, while considering the power costs of each sensor, and configures the network accordingly. In contrast to Impala and our system, MiLAN focuses on the quality of the sensor data only, but does not adapt other algorithms.

In the Pervasive Computing area, there are several examples for context-based adaptation. Gaia and its ad-hoc version Mobile Gaia [2] provide support for Active Spaces. Such an active space provides a single view to the underlying physical space by hiding the actual devices in its interior. Users and developers can use this abstract computing environment and do not have to address individual entities. Personal devices in the vicinity are automatically added to the personal active space whereupon the system loads additional components and runs services to use this new device. Gaia also implements a Model-View-Controller paradigm and adapters between these three entities that ease the development of applications running with different input and output elements. In contrast to our work, the clustering of Gaia does not aim at the adaptation of the system but is the reason for subsequent adaptation need. This adaptation is carried out to enable new devices but not to improve the performance of an algorithm.

In PCOM [1] as well, spatial proximity of devices defines the context. Applications are composed of components that may run on different devices. Contracts specify the dependencies between a component and other components or the underlying computing platform. If an application is started, PCOM selects necessary components according to the contract of the root component while respecting the resource limitations of the devices. In case of a network change, PCOM tries to reselect the dependency to another component to keep the application running. In contrast, our system does not deal with the adaptation of the application, but with algorithms and their interiors.

Several adaptive applications exist in the Pervasive Computing field. In general, an application can retrieve context information from a context model and adapts its behaviour based on that information. In [10], a user-adaptive and location-aware task planer is presented. The alarm time for a task is continuously recalculated based on the current distance between the user and the event. The purpose of tasks can also be specified (e.g. buying food) and the system enables the tasks based on the location and time of the user and the opening hours of places. Our approach adapts on a much lower level and is, therefore, orthogonal to such adaptive applications.

3. CONTEXT FOR GROUP FORMATION

As we have shown, context-based adaptation in Pervasive Computing focuses on the application level, typically by searching and (re)integrating services into the application or by changing the application behaviour or presentation. The information needed for such adaptation is "classical" context, i.e. the identity of entities, their location and the time of the information.

For adaptation of algorithm, context parameters have to be included that affect the behaviour of algorithms. Since nodes with similar context should be configured similarly, group formation has to consider the same context. This context can be divided into three categories: node hardware, network topology and application characteristics.

3.1 Node hardware

Computing devices used in Pervasive Computing and Wireless Sensor Networks can exhibit varying hardware capabilities. In most cases, RISC processors are used, but they range from 8bit Atmels with 8 MHz to 32bit ARM processors with more than 1 GHz. Program memory and main memory also differ in similar dimensions. Communication hardware can also be different, from small sensor nodes that only use Bluetooth to PDAs with powerful 802.11 cards. Devices with more powerful processors or communication hardware usually have a battery with higher capacity. Finally, the number and type of extra devices may also vary: sensor nodes usually carry several sensors like temperature, brightness, humidity or vibration, and, for example, GPS devices might be included in PDAs.

From an algorithm's point of view, this variety is not desirable since basic algorithms usually assume homogeneous hardware. Thus, similar devices should be grouped so that algorithms can easily take care of the specific constraints of the underlying platform. Groupings that exploit the heterogeneity, e.g. where a powerful device in each group can take over computation or communication intensive tasks, usually include more than one algorithm that are coordinated by the application. This is handled by above-mentioned service-centric adaptation systems for Pervasive Computing.

3.2 Network topology

Devices can either be static or mobile. Mobile devices have the ability to move itself (e.g., a car in the EMMA scenario), are carried (e.g., a PDA) or are attached to a moving object (e.g., in a Wireless Sensor Network scenario where this object is the observed one). We do not examine mobility in this work but focus on the static states that occur between mobile states. Therefore, mobility patterns are not taken into account for group formation.

Static devices can be deployed freely which happens either systematically or randomly. The latter is the often cited example of sensor nodes thrown from an airplane. Systematic placement can lead to very regular patterns, e.g. a grid which is useful to uniformly observe an area.

Networks of mobile or randomly placed devices have a varying density of devices. Only in regularly formed grid scenarios, the density is equal, except at the border. Devices that are more closely together often perform joint work as the applications in Section 2 have shown. Since algorithms are dependent on the density of nodes and should be optimally coordinated between devices that are interdependent, density is important grouping context information. Since the density can be calculated from the absolute positions of the devices, this belongs to "classical" context.

Closely related to density are constrained topologies. Usually, obstacles in the environment, e.g. walls in a building, separate devices from each other. In most cases, this separation has an analogy in the application. Therefore, adaptation of algorithms and group formation have to consider this. If physical separation cannot be deduced from density, the context model has to be queried. In this case, the devices cannot perform group formation and adaptation autonomously.

3.3 Application characteristics

While the last sections described physical properties of the devices and the network, this section focuses on software issues. Network traffic is such an important property since in wireless networks, large traffic loads on some devices can interfere with other devices. In peer-to-peer scenarios with equal devices, this is a minor problem. But in Wireless Sensor Networks with base stations where data is sent to or in the EMMA scenario where a data connector controls the data content between different hierarchical levels or in a Pervasive Computing application where a powerful device takes over many task from other resource-poor devices, traffic concentrates around certain points. Therefore, the network traffic has to be taken into account for group formation.

The fact that the traffic increases when the distance to such concentration points decreases can be exploited. Instead of the traffic load measurement, the distance of a device to such a point can be used. Since this requires location information, it is related to the second category discussed above.

4. GROUP MODELS

After we have explained the context necessary and relevant for group formation, this section gives an overview of the group formation models using the context in their metrics. Besides "used metrics", the model space consists of two more dimensions: "coordination" and "group layout". The complete model space is discussed in the following.

4.1 Model space

4.1.1 Coordination

The dimension "coordination" defines where the group formation algorithm is running. There are two extremes: running all computations on a central component or distributing the computation on all nodes.

The central approach allows to take the global situation into account which enables high optimisation possibilities. On the other hand, it requires to transfer the state needed for group formation via the network and distribute the created groups back to the network. Especially in large networks such an approach can result in a large overhead.

The distributed approach requires no large scale state transfer since the group formation is performed on each node in the network and uses only local information. Since all decisions are based on limited information no global optimisations are possible.

Combining the benefits of these approaches results in a hybrid method. The basic idea is to analyse the global situation by a central component. Based on this computation a distributed algorithm is configured. Since analysing the global situation does not require every nodes' state in detail, aggregation can be used to limit the amount of data to be transfered.

4.1.2 Used metrics

Grouping nodes requires the existence of a metric defining a mapping of nodes to groups. This metric is heavily dependent on the domain of the algorithms to be adapted. In case of routing algorithms the traffic load, neighbourhood and location of nodes are possible candidates.

4.1.3 Group layout

When using the formed groups to adapt algorithms, an adjustment between neighbouring 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 negotiations between every node of two groups would require too much communication.

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.

4.2 Models

The different models are briefly introduced now. Since the "coordination" dimension has the biggest influence on the model, they are ordered by this dimension.

4.2.1 Centralised approach

In the centralised approach, the groups are formed by a single algorithm running on the central coordinator. Based on this idea, the group formation is divided into three phases:

1. Collect node information and transfer them to the central coordinator
2. Create groups and assign cluster heads
3. Inform cluster heads about their members

Weighted Links - Graph Partitioning.

The *Weighted Links* (WL) approach 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 [3], the graph is divided in several subgraphs, which contain the nodes to be grouped.

The graph representing the network has nodes N corresponding to the nodes in the real network and edges E between nodes where communication between the nodes in the real network is possible. Each node is assigned a *node property value* based on the used metric, e.g. the number of neighbours or the traffic load (v_k, *node property value* of node k). Every edge e_{ij} (edge between node i and j) is weighted depending on the properties of the corresponding nodes using Formula 1. This formula assigns links between nodes with similar properties values near 1 and nodes with different properties values near 0.

$$e_{ij} = min(v_i, v_j)/max(v_i, v_j) \tag{1}$$

$$C(G) = \sum_{e_{ij} \in E \& n_i \in G_{l_1} \& n_j \in G_{l_2} \& l_1 \neq l_2} e_{ij} \tag{2}$$

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 minimising the cost function listed in 2. This function covers those edges, where the connected nodes are assigned to different groups.

4.2.2 Distributed approaches

Most Equal Neighbour.

Generally, the *Most Equal Neighbour* (MEN) approach is based on the idea that two neighbouring 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 neighbouring 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 neighbourhood. After that, every node knows its neighbourhood and their values. Based on the own and the neighbours' values, a node can weight the corresponding links and selects the neighbour with the most equal value to form a group with it. If two neighbours have the same values, the neighbour with the lower ID will be selected. Every node informs its selected neighbour. All selected edges between nodes are forming a spanning tree which can be used in the second phase to elect a cluster head for each group.

The used metric defines the *node property value* and, therefore, the weight of the link. In case of a location dependent metric every node can broadcast an arbitrary number since the receiving node can determine the closest neighbour (the location of both nodes is most equal) by comparing the RSSI (received signal strength indicator) values of the incoming packets. When using the neighbourhood metric, each node counts the number of neighbouring nodes. For this purpose, the existent network traffic can be used and, therefore, no additional traffic has to be generated. Based on this existent traffic it also possible to calculate the traffic load.

Distributed Groups - Static Table.

While the previous distributed approach builds connected groups, this approach results in distributed ones. The 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.

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 3, a second Mapping Function 4 exists, which maps the result of $f_{property}$ to a group identifier.

$$f_{property}(node) \rightarrow integer \tag{3}$$

$$f_{group}(integer) \rightarrow group\_id \tag{4}$$

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 would result 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 3 has to be a fixed interval, e.g. the interval between zero and one hundred. Without a fixed interval the second Mapping Function 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.

4.2.3 Hybrid approach

Distributed Groups - Dynamic Table.

When applying the hybrid approach to the *Distributed Groups - Static Table* (DGST) model, the static table is replaced by a dynamic table (*Distributed Groups - Dynamic Table*; DGDT). The dynamic table is calculated by a central component. In contrast to the centralised approaches, it is not mandatory to transfer the total state of each node to that component. In the most trivial case only the minimum and maximum value of the property is transfered, which enables the possibility of aggregation, to limit the required traffic. Based on these two values, an optimised table can be created. Since the table is adapted to the values which are actually present in the network, this model can also be used for metrics where it is impossible to define reasonable upper bounds for the measured node properties. 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.

Location Based.

The group formation process of the *Location Based* model (LB), 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*[12, 11], 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 neighbourhood 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 every standard clustering algorithm 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 such messages are used where the identifier of the message and the own virtual cluster head are equal.

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.

In scenarios where data is routed to a central coordinator, the traffic processed by a node increases with the closeness to the coordinator. Based on the assumption that the behaviour 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 central component. Using only the distance property to form groups results in small groups in the area around the central coordinator and large groups with high in distant areas. The *Circular Pattern* arranges the virtual cluster heads, see Figure 1, on circles around the central component, which results in equal sized groups.

4.3 Model rating

When analysing the different models, it becomes obvious that a completely centralised algorithm does not scale with an increasing number of nodes. Since the *WL* algorithm weights the edges between nodes the properties of the individual nodes have to be known to the central coordinator which prohibits the usage of aggregation. Without aggregation the amount of state that have to be transfered is high. Another problem of this approach is the fact that graph partitioning is a NP-hard problem, even if there exists algorithms to approximately calculate the groups with less effort, the formation of the groups requires considerably

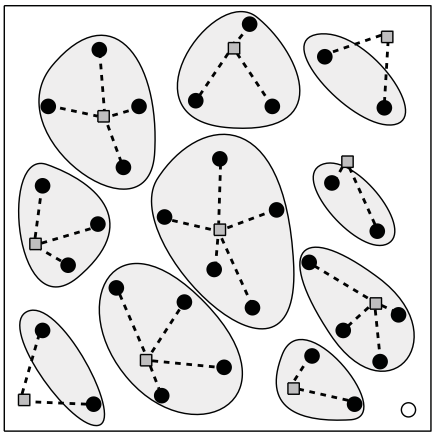


Figure 1: Grouped nodes (■) using virtual cluster heads (■) arranged by the *Circular Pattern* around the sink (□). Nodes are assigned to the minimal distant virtual cluster head (- -).

effort. Due to the high costs of this approach we do not consider the method of group formation in the following.

Using the *DGST* method requires minimal effort to form groups. All nodes can calculate the group they belong to using only local information which they can compute by evaluating the background traffic. Whenever a node overhears a message from another node, it can store the id of the node. Counting the collected distinct IDs results in a approximation of the neighbourhood size. Since no global knowledge is used this approach cannot be used with an arbitrary metric. And even with a suitable metric, like the one based on neighbourhood, this approach cannot give any guarantees about the groups sizes. Considering these drawbacks brings us to the conclusion not to further investigate this approach.

The main advantage of the *MEN* approach is the fact it can use an arbitrary metric to form groups distributed. However, since the formation is based on the idea of selecting the most equal neighbour the sizes of the created groups are probably small. Even if the size of the formed groups can be adapted by using also the second equal neighbour, if it is similar to the most equal, it is difficult to select an appropriate threshold. Small groups have only limited advantages in comparison to the case where all nodes are configured individually. Even if this approach promises some interesting features we do not consider it in the following.

Dynamic tables, introduced by the *DGDT* approach, allows to use a wide spectrum of metrics because it removes the demand to map the node properties uniformly to a predefined interval. Even if the central coordinator needs global knowledge the amount of state to be transfered is limited because the state can be aggregated during the collection. Since the density of nodes is an important factor when adapting the transmission range of routing algorithms, in the following we focus on *DGDT-N*, where *N* denotes the used metric (Neighbourhood).

Another promising approach is *LB*. Since this method is also using the idea of a centrally configured distributed algorithm, this approach is able to adapt to the concrete network situation without the requirement of transferring a lot of state which would result in a not scalable solution. Even if this approach allows to use sophisticated methods to create

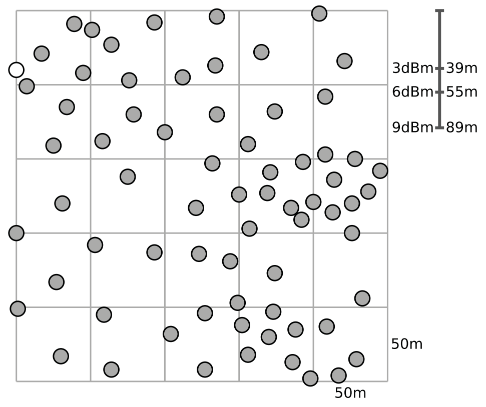


Figure 2: Layout of the cooperating objects scenario: nodes (■) and data sink (□).

the pattern of the virtual clusterhead locations, we focus on a regular pattern to reduce the amount of state to be send through the network. In the following we use *LB-C* as an abbreviation for location based model with a circular pattern.

5. EVALUATION

After having selected two interesting group formation models, their quality is evaluated. This is done by rating the performance of the adapted network configuration. Since every model introduces some overhead, for example 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. Therefore, it is compared with an established lower and upper bound.

5.1 Experimental Setup

5.1.1 Scenarios

To evaluate the grouping models we introduce two scenarios. The first one used during the evaluation of the group formation approaches models a Wireless Sensor Network (WSN) and consists of 100 nodes. These nodes are arranged regularly on a grid with an edge length of ten nodes with an distance between the nodes of 30 meters. In the second scenario, the nodes are randomly placed as in typical Pervasive Computing scenarios. To model areas with a high node density like the engine compartment in the EMMA project the nodes are not uniformly distributed. The positions of the nodes is visualised in Figure 2. In both scenarios the data is routed to a sink which is located in the top left corner.

For 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. In the Wireless Sensor Network scenario, nodes using the lowest setting can only reach the four nearest neighbours, whereas 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 neighbourhood to 20 nodes.

The course of the scenario is structured as follows:

- **Initialisation Phase** ($0s - 300s$): The routing algorithm constructs a routing tree. The *TrafficGenerators* create data and send them to the central coordinator. Every node is configured with a transmission power of $6dBm$.
- **Group Formation Phase** ($300s$): The group formation component creates the groups using information about the node positions and neighbourhood sizes. The information is collected during the *Initialisation Phase*. Since the focus is on the quality of the created groups, we abstracted from the state transfer and using shared variables to access the required information.
- **Adaptation Phase** ($300s$): In this phase the nodes are adapting their transmission power using the formed groups.
- **Reconfiguration Phase** ($300s - 600s$): During this phase, the routing tree is reconfigured, i.e. new parents are selected. Due to the adapted transmission power, new neighbourhoods are formed. The *TrafficGenerators* are disabled during this phase.
- **Test Phase** ($600s - 1200s$): After enabling the *TrafficGenerators* data is routed to the central component.
- **Evaluation Phase** ($1200s$): During this phase the total amount of data received by the central coordinator 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).

5.1.2 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. 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 by the user this data has to be sent to the second type of nodes, the *Coordinator*, which simulates the node with the user interface where the data is required. The traffic generator at the sending nodes creates packets with a size of 30 bytes. The traffic generated by one node is on average three bytes per second, which results in sending one packet every ten seconds.

Since multihop communication is required to send data to the coordinating node a routing algorithm is used. The underlying concept of this algorithm is based on the construction of a spanning tree. The coordinator acts as the root of this tree. The data is forwarded to the root by sending the data to the parent node in the tree. 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 and, therefore, in the following the algorithm is explained in detail.

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 root and a list of neighbourhood nodes is sent per broadcast. Since the space in the beacon is limited only ten neighbourhood nodes are included. The selection of the ten nodes is based on the time when the node was detected for the 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 neighbourhood 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 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 neighbourhood 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 neighbourhood 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 stale 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.

The routing algorithm requires the transmission of packets to neighbouring nodes. This single hop communication between two neighbouring nodes is controlled by the *IEEE80211* component, which is part of the network simulator. Since unicast packets are transmitted using acknowledgements lost packets are automatically retransmitted. The maximum number of retransmits is set to 3.

The evaluation of our models requires a network simulator allowing to change simulation parameters, like the transmission power of a node, at runtime. Since this demand is essential for the our evaluation approach, see next section for details, we are using the *CUBUS* network simulator, which was developed in our department. *CUBUS* is a discrete event simulator. The used object model allows set or change the configuration of every component, for example the protocols in network stack or the network interfaces statically using a scenario definition file or at runtime using a command line interface. Due to its modular design it is is possible to exchange implementations of protocols at runtime.

5.2 Lower and upper bounds

When configuring every formed group with equal parameters, no differences between using and not using groups are visible. For this reason, the performance of an optimally configured grouped network is higher than using no groups and, because of this, using no groups acts as the lower bound.

As a second step, the upper bound is determined. 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 at least 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.

Due to the large space of possible network configurations (3 settings to the power of 100 nodes results in about 10 to the power of 50 configurations) it is infeasible to simulate and rate all of them. Therefore, a genetic algorithm is used during the investigation for an optimal configuration. The behaviour 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 approximated. To improve this process, this search 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 since an optimal model groups exactly those nodes with equal settings.

Using the genetic algorithm, discussed in the previous section, we found the optimal parameterisation for both scenarios. Figure 3(a) and Figure 3(b) shows the determined transmission power for the nodes of the sensor network and cooperating objects scenario. Keeping the location of the sink of the data in mind, it attracts attention that in both scenarios the nodes around that sink are configured with an high transmission power. In case of the sensor network scenario the nodes at the top and left border are also configured with the maximum transmission power. When investigating on the reason for this configuration it turns out that the nodes at the borders are acting as attractors for the data. The data is routed from the centre of the network to the borders and from there to the sink. Since the nodes at the border have large sending range the path using those nodes results in minimal path lengths.

Analysing the configuration of the cooperating objects scenario shows a clear interrelation of the density and the optimal configuration. The areas at the top and the left side of the network are configured with the medium transmission power. High density areas on the right and bottom right side of the network are configured with an low transmis-

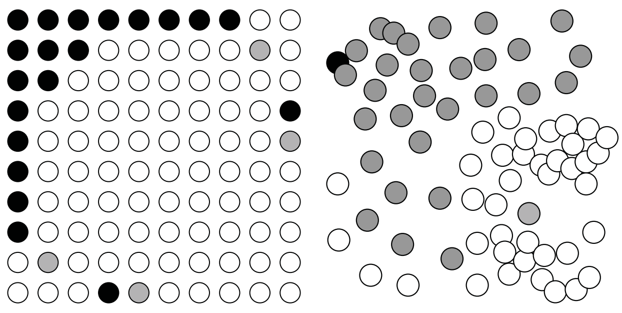


Figure 3: Optimal node configuration for both scenarios (■ 9dbm ▨ 6dbm □ 3dbm)

sion power. The interrelation is meaningful because tuning the transmission power is a tradeoff between the number of hops that are required to route a message to the sink and the number of collisions on these paths. During the search for an optimal configuration it turns out that the configuration of a few nodes in the network are crucial for the overall performance of the network. The reason for that is the fact that configuring these nodes with an appropriate transmission power creates links between two areas which results in completely new paths to the sink and, therefore, the load on all paths is reduced.

As discussed at the beginning of this section we are using the equally and optimal individually configured nodes to evaluate the performance of our group formation models. When setting up the network with equally configured nodes it turns out that the medium transmission power provides in best results. The average target ratio and the distribution for several simulation runs for the equally and optimal individually configured nodes are visualised for the sensor network scenario in Figure 4. Figure 5 show the same values for the cooperating objects scenario. When comparing the curves the huge potential for the group based node adaptation becomes visible. In case of the sensor network scenario the average target ratio can be increased from 54.5% to 84.5%. Even if the difference in the cooperating objects scenario is smaller a potential increase of 15 percentage points is possible.

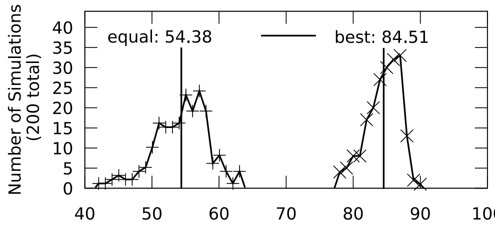


Figure 4: Target ratio for equally and optimal individually configured sensor network scenario.

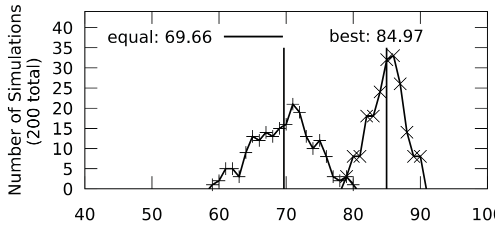
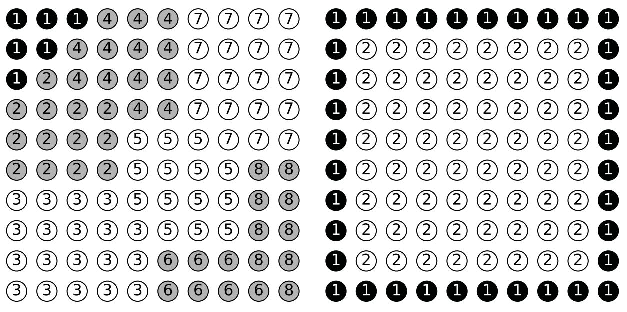


Figure 5: Target ratio for equally and optimal individually configured cooperating objects scenario.

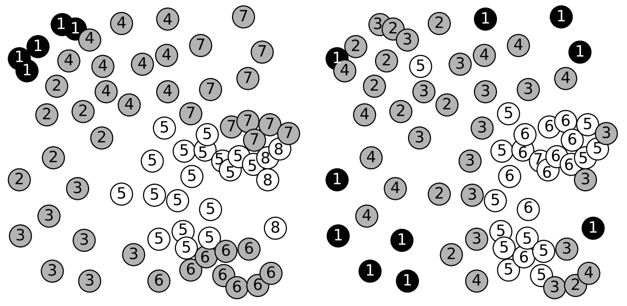
5.3 Group configuration

Knowing the upper and lower bound, in the following step the performance of the networks adapted by the group formation models LB-C and DGDT-N are evaluated. For this purpose, the models are applied to the network scenarios and groups are created. Even if the number of possible configurations is many magnitudes smaller than those of the individually configured nodes (8 groups with 3 possible settings results in about 10^4 configurations), the number is still too large to simulate them all. Therefore we use the genetic approach again.



(a) LB-C (b) DGDT-N

Figure 6: Node configuration for sensor network scenario (■ 9dbm ■ 6dbm □ 3dbm)



(a) LB-C (b) DGDT-N

Figure 7: Node configuration for cooperating objects scenario (■ 9dbm ■ 6dbm □ 3dbm)

The groups, formed by the two models, are visualised in Figures 6(a) and 6(b) for the sensor network and in Figure 7(a) and 7(b) for the cooperating objects scenario. The numbers in the circles are representing the group ID, i.e. nodes with equal IDs are in the same group. In case of the LB-C model the groups are arranged in circles around the sink. Group 2 and 4, for example, are building the second and Group 3, 5 and 7 the third ring around the sink. When analysing the groups formed by DGDT-N in the sensor network scenario, showed in Figure 6(b), it turns out that only two groups are formed. This behaviour is expected because due to regular layout of the nodes there are nodes at the border with low and nodes in the middle with high density. Actually the 4 corner nodes have a different neighbourhood than those at the border, but the model tries to create equal sized groups and therefore the border and corner nodes are merged to one group. In the cooperating objects scenario the DGDT-N groups the nodes at the borders and creates several groups for the high and low density areas.

Figure 6 and 7 show the parameterisation for the formed groups using black, gray and white filled circles for nodes with $9dbm$, $6dbm$ and $3dbm$ transmission power, respectively. When analysing the formed groups in the sensor network scenario huge differences between the two models become visible. While the LB-C uses a schema where nodes with increasing distance to the sink are configured with a decreasing transmission power and the most distant nodes with the medium power value the DGDT-N model uses a completely different schema. Here the nodes in the centre are configured with the minimum transmission power and the nodes at the edges with the maximum. When comparing the configurations with the optimal individual one, it turns out that the configuration based on DGDT-N is very similar to the optimum. In the cooperating objects scenario the differences between both models are minimal. Each model configures the nodes depending on the density. Like in the optimal configuration, areas with a high density are set up with a low power value and areas with a low density with a high transmission power.

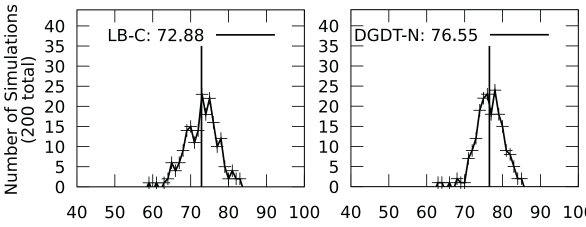


Figure 8: Target ratio for LB-C and DGDT-N grouped sensor network scenario.

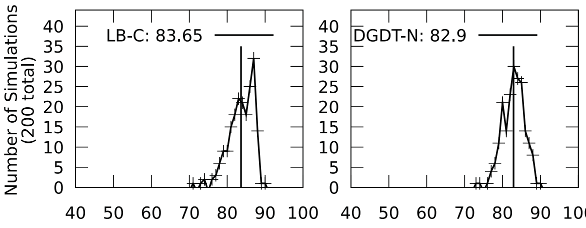


Figure 9: Target ratio for LB-C and DGDT-N grouped cooperating objects scenario.

After having compared the group configuration with the optimally configured single node configuration, the performance of the resulting networks is presented. Figure 8 shows the distribution of the target ratio of 200 simulation runs for each model in the sensor network scenario. In case of the LB-C model on average 72.9% of the data reaches the sink. Using the DGDT-N model the average target ration is 76.6%. Comparing these values with the equally configured network, we can increase the target ratio about 34% or 41% respectively. The results for the cooperating objects scenario are visualised in Figure 9. Using the LB-C model we able to increase the performance of the network up to a target ratio of 84% which is 1 percentage point below the maximum when adapting every node individually. Using the DGDT-N model the performance of the network can be increased about 20% in compare to the equally configured network. With a target ratio of 83% DGDT-N has a minimal disadvantage in this scenario.

5.4 Discussion

Based on the 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 sink and nodes around it are set up

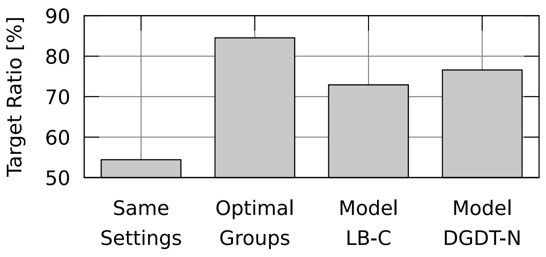


Figure 10: Results for target ratio for sensor network scenario.

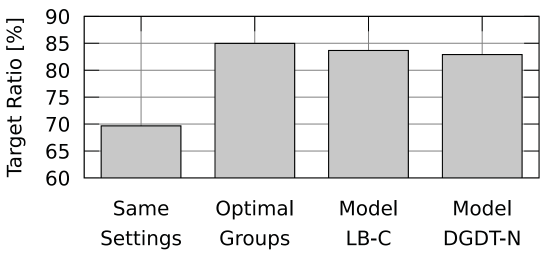


Figure 11: Results for target ratio cooperating objects scenario.

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 minimise the number of collisions it is useful to minimise 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 maximising 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 with 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 higher 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 neighbourhoods of the nodes are equalised and interference is diminished.

6. CONCLUSION

This paper has shown that adaptation is useful and possible also on lower levels to increase the performance of the system. To perform it, classical context information is used, for example location, but also new context information is needed that goes beyond the classical context used in Pervasive Computing applications like network traffic.

Adaptation does not have to be global to achieve good results, but can be done more locally, i.e. in groups of nodes. These groups are also formed based on the extended context. We have identified several possible models to form groups suitable for adaptation. Evaluation then showed that the performance can be significantly increased and can reach almost the optimum.

From the evaluation results, general rules have been deduced based on the context. These rules make it possible for general adaptation framework like TinyCubus to adapt grouped networks with a different network setting and different number of groups.

Our next steps are the inclusion of the group formation models and the adaptation rules in our general adaptation framework TinyCubus to ease the test with other scenarios. Then, we plan to test the combination of TinyCubus with an adaptive systems working on application level to show that the whole system can profit from the low-level adaptation, but also from adaptation on application level.

This analysis focused on an routing algorithm only. It is necessary to test the group formation with other algorithm classes, too. To ease the finding of the optimal configurations and the deduction of rules, an automated mechanism has to be developed exchanging the manual steps of the experiments.

7. REFERENCES

[1] C. Becker, M. Handte, G. Schiele, and K. Rothermel. PCOM - A Component System for Pervasive Computing. In *Proceedings of the 2nd IEEE International Conference on Pervasive Computing and Communication (PerCom 04)*, 2004.

[2] S. Chetan, J. Al-Muhtadi, R. Campbell, and M. Mickunas. Mobile Gaia: A Middleware for Ad-hoc Pervasive Computing. In *Proceedings of IEEE Consumer Communications and Networking Conference (CCNC 2005)*, January 2005.

[3] U. Elsner. Graph partitioning - a survey. *Preprintreihe, TU Chemnitz*, SFB393/97-27, December 1997.

[4] Embedded Middleware in Mobility Applications (EMMA) project web site. http://www.emmaproject.eu.

[5] W. B. Heinzelman, A. L. Murphy, H. S. Carvalho, and M. A. Perillo. Middleware to support sensor network applications. *IEEE Network*, 18(1):6–14, 2004.

[6] T. Liu and M. Martonosi. Impala: A middleware system for managing autonomic, parallel sensor systems. In *Proc. of the 9th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming*, pages 107–118, 2003.

[7] P. J. Marrón, A. Lachenmann, D. Minder, J. Hähner, R. Sauter, and K. 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.

[8] P. J. Marrón, D. Minder, and Embedded WiSeNts Consortium, editors. *Embedded WiSeNts Research Roadmap*. Logos, 2006.

[9] D. Minder, P. J. Marrón, A. Lachenmann, and K. Rothermel. Coordinated group adaptation in sensor networks. In *6th Fachgespräch Sensornetze, Technical Report AIB 2007-11, RWTH Aachen*, pages 43–46. Distributed Systems Group, RWTH Aachen University, July 2007.

[10] C. Stahl, D. Heckmann, T. Schwartz, and O. Fickert. Here and Now: A User-Adaptive and Location-Aware Task Planner. In *Proceedings of the International Workshop on Ubiquitous and Decentralized User Modeling (UbiDeUM'2007)*, pages 52–63, June 2007.

[11] J. Zhao, R. Govindan, and D. Estrin. Computing aggregates for monitoring wireless sensor networks. In *Proc. of the First IEEE International Workshop on Sensor Network Protocols and Applications*, 2003.

[12] 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.