Map-based Support for Wireless Sensor Network Simulation

Piotr Szczytowski, Abdelmajid Khelil, Neeraj Suri
Technische Universität Darmstadt, DEEDS Group, Germany
Hochschulstr. 10, 64289 Darmstadt, Germany
Email: {szczytowski, khelil, suri}@informatik.tu-darmstadt.de
Tel: +49 6151 16 {3414, 3414, 3513}
Fax: +49 6151 16 4310

***Abstract*— Wireless Sensor Networks (WSN) are receiving growing attention in the research community. As simulation is a frequently used approach also to test and validate approaches, the simulation environments need to be able to support the evolving WSN design schemes. While a growing research trend in WSN is to address regions instead of single sensor nodes, existing WSN simulation environments still do not support modeling and design on this abstraction level. In this context, we propose MAP++ as a framework that extends the OMNeT++ network simulator. It provides a suite of tools needed to support WSN simulations. In particular, MAP++ supports map-based topology and scenario generation and evaluation. It also provides trace data visualization and database powered analysis. The usability of MAP++ is illustrated via examples of applications over each phase of simulation development, i.e., modeling, design, implementation and performance evaluation, and case studies for map construction algorithms. We show the utility of our tool chain to enhance the common statistics-based trace analysis, and to conduct algorithm evaluation and comparison.**

***Index Terms*— Wireless Sensor Networks, Simulation, Map, OMNeT++**

I. Background and State of the Art

Wireless Sensor Networks (WSN) entail rapidly developing concepts in the fields of communication and computing. The progress in sensor miniaturization is increasingly leading to advanced monitoring/processing capabilities. The increasing computing and communication capabilities provide growing flexibility through self-* capabilities. For instance self-organization allows the sensor nodes to be randomly deployed by distributing them aerially over the target area. Overall, the scope of WSN applications is growing very rapidly. Unfortunately, the energy constrained lifetime with the main drain incurred by wireless communication is still a major limitation for WSNs.

The inherent redundancy of sensor nodes and the spatial nature of the monitored physical phenomena results in spatially correlated sensor readings, and indirectly also the network properties such as energy distributions. The natural abstraction to capture the inherent spatial correlation of sensor nodes states is the map paradigm. A *Map* is an aggregated view on the spatial distribution of a certain attribute at a certain time. The literature describes two main classes of maps, i.e., the choropleths [1] and the isomaps [2], [3], [4]. The construction of the map visually groups spatially correlated sensor nodes with similar attributes values into *regions*. We define a region by its border (a set of spatial points) and an aggregate (e.g., average) of the attribute's values obtained from all sensor nodes located in the region's area.

The map paradigm builds on the region principle and therefore provides excellent modeling primitives for WSNs. Global maps are created for the sake of network monitoring (e.g., residual energy map [5]) or of event detection (e.g., oxygen map [6], [7]). A promising direction consists in using maps to optimize protocols [8], [9], detect [10] or track [11] event boundaries. These papers highlight the map-based methodology as a powerful and highly promising abstraction level. In [12], [13], we propose the maps as the natural step towards a holistic *Map-based World Model* and *Map-based WSN design*.

In most network simulators, both the design of topologies and the storing/representation of the trace data have textual format. The expression of the spatial dependencies in that form is both complex and limited. WSNs usually involve large scale simulations, where manual preparation of the multitude of topologies and simulation scenarios is beyond simple textual representation. The common solution to this problem is using text generation scripts that create topologies following the simulator syntax. Unfortunately, techniques to verify and validate the created topology are limited. A much more direct and effective solution is doing visual validation.

The established OMNeT++ [14] simulation environment introduces the notion of the interactivity by offering a visual editor for its topology format. However, topology is only one feature of scenario generation. In WSN further important features such as temperature's spatial and temporal distribution present a fundamental part of scenario generation. OMNeT++ does not explicitly support the generation and visualization of attributes of the network and the physical world. The lack of support for visualization is even more apparent in the case of trace data. Currently, trace data is collected in a text file used for statistical analysis. Usually, the analysis is completed at sensor node or event level. This is tedious and does not

Research supported in part by DFG GRK 1362 (TUD GKmM), EC INSPIRE, EC CoMiFin.

follow the region abstraction level as discussed earlier. Alternately, visualization rendered as a map is a natural approach for the discovery of spatial dependencies. The map representation cannot fully replace the traditional statistical approaches, which are more appropriate for understanding the details of the concepts under investigation. However, the map-based simulation approach offers considerable support for high-level designing and debugging. The processing of data across series of simulations requires the efficient extraction of relevant information from complex traces with mixed content. The efficient way of handling the large amount of data is to export it to a relational database to process using a database engine.

In this paper, we extend our preliminary work [15]–[17] and provide a map-based MAP++ framework that provides the following contributions:

- Map-based topology generation support that simplifies the modeling and scenario generation,
- Map-based trace data visualization support that identifies spatial dependencies for an efficient design and debugging,
- Storing and processing of map-based trace data using relational databases for an easy analysis and performance evaluation.

The contributions are highlighted by set of case studies. In these case studies we evaluate set of algorithms representative for WSNs to show the benefits of MAP++ framework.

The remainder of the paper is structured as follows. In Section II, we briefly specify our system model. In Section III, we review the existing simulation environments for WSN, outline their limitations and state our objectives. Section IV details the architecture and implementation of our MAP++ to fill the gap of existing simulation environments. In Section V, illustrations of MAP++ application scenarios are presented. Section VI investigates the framework performance and highlights the usability of MAP++ through two representative case studies.

II. System Model

We consider a two dimensional WSN consisting of stationary, resource-constrained sensor nodes with limited processing, storage and battery-capacity. Each sensor node is capable of short range wireless communication with a fixed transmission range. We focus on large scale deployments involving hundreds or thousands of sensor nodes. A sensor node measures a physical signal (temperature, humidity, etc) or its status (e.g., its own battery status) at a specified sampling rate. There also exists one or more base stations (called sinks), which are designated as resource-rich nodes serving as a gateway between the WSN and users. We also allow for scenarios, where sensor nodes are equipped with different set of sensors, monitoring different phenomena.

III. WSN Simulation Environments

The increasing interest in the WSN research has resulted in the development of a variety of simulation environments. TOSSIM [18] as a simulator for the common TinyOS platform [19] serves as a good example of such environments. Its attractive feature is that code developed for the simulator can be directly deployed on real sensor nodes running TinyOS. The simulation is very accurate on the node architecture level. Unfortunately, these architecture details (e.g. module wirings) consume much of simulation processing resources while providing only limited impact on global properties of WSN. Consequently, TOSSIM's high level of details limits the simulated network scale, rendering the simulator inapplicable for large scale scenarios.

In addition to the development of new simulators, some of the existing network simulators have been extended to support WSN simulation. For instance, the established NS-2 simulator [20] has been extended by the MannaSim Framework [21] through developing a topography generation tool, modules simulating an antenna, and radio propagation models. However, the acceptance of NS-2 in the WSN community is limited due to its poor scalability [22].

Further projects like [18], [23]–[28] deliver specific libraries of components and algorithm templates leveraging different aspects of WSN. Some of these tools offer various forms of visualization. Unfortunately, the existing libraries operate on message level and not on region/map level.

OMNeT++ [14] also follows the approach of adapting its features to use for simulation of wireless networks. OMNeT++ is a component-based, modular, and event discrete simulator. It is based on a generic and flexible open-architecture, provides extensive graphic user interface and is platform independent. These characteristics make it especially suitable for highly efficient network simulations. Owing to its modularity, and consequently extensibility, it is receiving a growing usage in the WSN community resulting in the implementation of mobile and sensor frameworks for this simulator. For instance, the Mobility Framework for OMNeT++ [29] is intended to support wireless and mobile simulations using OMNeT++. It provides features such as sensor node mobility, dynamic connection management and a wireless channel model. A similar approach is presented in MiXiM [30], which is prepared as a simulation framework with a concise modeling chain for mobile and wireless networks. While not yet implemented, it promises to deliver models for mobile environments, sensor nodes and objects, radio propagation models for multiple signal dimensions, physical layer models for modulation, coding and diversity receivers, library of Medium Access Control (MAC) protocols and localization algorithms. Similarly, [31] extends OMNeT++ by an advanced radio/channel model allowing multiple transmission power levels, a physical process model and a resource monitoring MAC protocol.

The authors of *EYES WSN Simulation Framework* [32] and *NesCT* [33] investigated the possibility to merge TOSSIM and OMNeT++ and provided necessary support. The developed simulation environment profits from rich

TinyOS libraries as well as the increased efficiency and extensive visualization provided by OMNeT++. EYES uses maps for defining the failure probabilities of deployed sensor nodes or marking the radio propagation obstacles. However, EYES does not support generalized maps. [34] presents a tool-chain for providing the parametrization, distributed execution and result-postprocessing and debugging. Similar to our approach, [35] provides offline visualization of traces. However, it works on the node-level basis, not utilizing the facets of spatial correlation of sensor nodes attributes. [35] is limited to offline replaying of simulations and does not provide support for scenario generation and evaluation.

The multitude of mentioned works confirms the rising interest in developing the simulation environment for WSN. Unfortunately, any project within the WSN community does not provide holistic support for map design and modeling in simulation environments. Approaches in other communities such as Space Time Toolkit [36] provide advanced capabilities for integrating spatially and temporally-disparate data within 2D or 3D display domain. The lack of access to their source code renders their integration with WSN simulations systematically difficult. To the best of our knowledge we are the first to provide a complete open source tool chain that considers the inherent spatial correlations in WSN.

IV. The MAP++ Architecture

In the following, we describe the main elements of the MAP++ framework architecture, their interdependencies and interactions with the OMNeT++ simulator.

A. Overview of the MAP++ Architecture

Fig. 1 illustrates the MAP++ framework architecture and the interactions with the OMNeT++ simulator. Simulation studies usually require a large number of scenarios to evaluate a wide spectrum of parameters. The *Scenario Generator* is responsible for the generation of the topologies designed by the user and supplemented by simulation scripts for batch execution using varied parameters. In order to provide generic solutions that could be successfully combined with different simulation engines, the framework requires a flexible and robust way of tracing the data. The generation of trace should be decoupled from the implementation of the simulation modules. Therefore, we design a separate module, *Trace Module*, which is independent from the rest of simulation and delivers a well formatted *Trace*. To simplify the process of configuring and embedding the Trace Module into the simulation, the *Trace Module Configurator* was developed. It provides the user interface, which allows seamless setting of *Trace Module* parameters. The OMNeT++ simulator executes the defined scenarios and embedded trace modules produce the traces. Traces are stored in XML format, which makes them suitable for automated parsing and map-based representation. The core functionality of representing the network using maps is realized through the *Visualization & Regioning* module. Visualization renders the map reflecting the trace data and allows basic map operations. *Regioning* implements additional map operations by providing the grouping of the nodes based on their class membership. A class is defined as a value range of the selected map attribute. More accurate and statistical evaluation of trace data is provided through the *SQL Database Interface* module. It allows the import of XML formatted trace data into a relational database and the querying of this data using a database engine. The query creation is simplified by the integration of the database module with the visualization module and as result allowing the users to visually create queries. Instead of manually typing the names of the sensor nodes to which the query should apply, user can select the appropriate for the query sensor nodes using the map.

From the implementation viewpoint, the framework consists of two distinct components, i.e., the Trace Module and the MAP++ Tools.

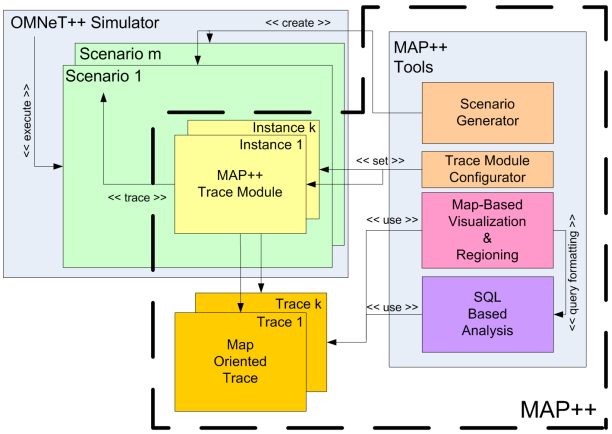


Figure 1. MAP++ Architecture

B. The MAP++ Trace Module

The Trace Module is embedded into the simulation environment as an instance of a simple module (object class implementation unit) defined within the topology file. The module is loosely coupled with the simulation environment, imposing only limited amount of additional work on simulator users. Its task is to periodically (period can be arbitrary set by the user) query all simulation modules of defined classes and store their selected published attributes. We developed the Trace Module with two objectives in mind. The first objective is for the Trace Module to be independent from the implementation of the simulation modules, as stated in the architecture overview. The developer of the simulation modules does not need to integrate the necessary traces into the modules source code. The simulation module should only publish the attributes required to track its state. This approach allows to use the MAP++ framework even with already existing simulations without need of making simulation code changes.

The second objective is to maintain consistent and reusable trace format. The traces format has to be consistent for proper interpretation for visualization by the MAP++ framework and their proper processing. We decided to use the XML format, that fully meets these requirements. The format's self description characteristic allows for automated import into relational database and processing. An additional benefit is the simplified conversion using stylesheet transformation to the desired format for use with a wide range of tools.

In case of WSN scenarios, with multiple map attributes of interest a separate instance of the Trace Module can be initialized for each module class, allowing parallel tracking of different sets of network perspectives. For example, if the network contains two different classes of modules each equipped with different sensors (e.g. temperature and humidity sensor nodes) then two instances of the Trace Module are need (Fig. 1, k = 2), as each Trace Module may trace only one module class. In case that a single module class provides two or more sensors, then only a single Trace Module is necessary for tracking.

C. The MAP++ Tool-Chain

The second component constituting the MAP++ framework is a suite of tools for generation of simulation scenarios, configuration of traces, visualization and analysis of simulation results.

Scenarios consist of topologies and batch scripts. The static topologies are created by randomly assigning the location for a specified number of sensor nodes within the defined deployment area. The links are established between the sensor nodes whose distance is shorter than the transmission range. Topologies and sensor nodes deployed in topology publish set of attributes, which are varied using batch scripts. Scenarios are stored in XML format and transformed by the provided stylesheet to comply with the OMNeT++ file format. The use of stylesheet transformation allows to adjust to the possible future changes in the file format and provides means for reusing the scenarios in different simulators for possible comparative studies.

The visualization is created by loading the topology definition and creating its two dimensional representation. The area occupied by nodes is fragmented into Voronoi [37] polygons, bringing the additional benefit of reflecting the sensor nodes density (Fig. 2). The individual polygons are filled with the shades of grey corresponding to the value of the selected attribute.

The results analysis is conducted using a relational database. The XML formatted trace allows automatized import of the results. Pre-parsing of the data allows the automation of the queries generation. The integration with the visualization provides means for spatially correlated queries, where queries are applied only to the nodes within a selected geographic area.

V. MAP++ Benefits and Applications

We first present the main functionalities of the framework at the different stages of the simulation process.

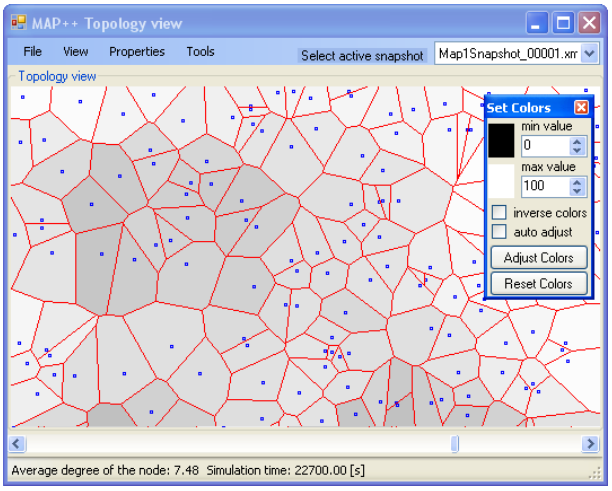


Figure 2. Voronoi-Based Map Visualization

Next, we describe how these functionalities can be applied. For this purpose, we present the main usage scenarios, i.e., modeling, design, debugging and performance evaluation. To better illustrate the frameworks capabilities and demonstrate its practical utility, we present two case studies in Section VI.

A. Overview of Applications and Benefits

We identify three different stages of the simulation process: Modeling, execution and results analysis. In the following we present the MAP++ benefits at each stage.

In the modeling stage, MAP++ supports (a) the definition of the modules, (b) the generation of the topologies (that use the defined modules), and (c) the creation of scenario sets (on basis of the generated topologies). The definition of a new module consists of the implementation of a new module class and defining the set of class attributes. The implementation of attributes on the developers side is limited to updating their value inside the module source code. The topology generation results in an automated creation of the topology files. Scenario sets correspond to producing the initialization files with sections describing varying values of published attributes. The initial conditions (initial values of published attributes) can be visualized and effectively defined (spatially correlated initial values) using the visualization tool.

Regarding the execution stage of the simulation, our framework currently offers assistance in the form of the Trace Module. However, we envision extending the usability of the Trace Module in future. Conceptually, the Trace Module could evaluate the maps at run-time. That capability would allow definition of conditions to which a set of actions could be assigned. For example on detecting a certain event the frequency of taking snapshots could be tuned or snapshots instead of being periodical become event triggered. Further extensions would include the real-time data visualization and streaming of traces into a database.

The MAP++ support for result analysis consists of the visualization and data processing using relational database. The visualization is projected as a two dimensional map, filled using shades of grey corresponding to the value of the selected attribute at a given time instant of the sensor nodes occupying the area. Basic operations such as the creation of differential maps are supported. Scrolling along the time axis displays the progress of the simulated phenomena. The simulations result in the creation of a large set of data. There is a need for extracting "significant" data from a large set of less relevant data. Our framework provides a solution for this problem by allowing the import of stored trace data into a relational database (Fig. 3). The use of structured queries allows an easy extraction of data of interest. As the trace data is imported off-line, the use of database engine does not generate the overhead while executing the simulation. Current implementation is provided for Microsoft SQL Server and MySQL open source database.

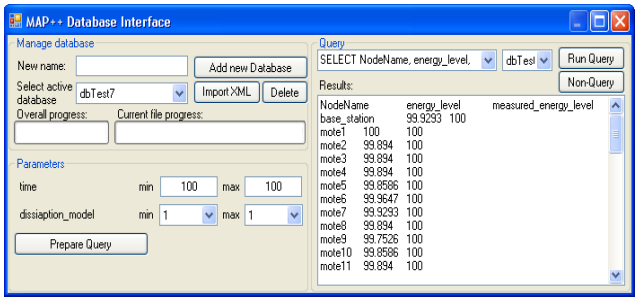


Figure 3. Database Interface

B. Modeling Stage

We illustrate the modeling capabilities of our tools through the example of energy consumption modeling. As a result of the WSN inherent sensor nodes redundancy, the energy depletion tends to show high spatial correlation. The sensor nodes located in close proximity of the sink deplete their energy at a much faster rate [38], [39]. This is due to the fact that besides their own operations, these nodes forward most of the WSN traffic. Another case of spatially correlated energy depletion is the spatial nature of most of the monitored physical phenomena. In the regions where the phenomena shows high activity, the rate of sampling and transmission of samples usually is increased to meet the required accuracy of monitoring, resulting in an increased energy usage ratio.

The modeling starts with generating the topology corresponding to the assumed system model. Our framework allows the setting of the communication range, size of the sensor field, number of sensor nodes and topology type (grid or uniform random). The creation of the scenarios containing mixed types of sensors is supported by iteratively adding new module classes to existing topologies.

The topology generation is followed by the accurate modeling of the phenomena to be monitored by the WSN. As physical phenomena usually show spatial correlations, the natural approach to model them is the generation of the corresponding maps. The iterative generation of maps for different scenarios (e.g. various distribution models) followed by their comparison enables precise modeling. MAP++ partially automates this process. Varying the values of the topology properties allows batch generation of scenarios. The visualized trace data simplifies the investigation of their impact.

The form for designing batch execution currently supports three types of properties: Numerical, boolean and string. In case of numerical values the user defines the minimal, maximal values and the step (value by which the parameter should be incremented) for generating values range. For the string properties, the creation of a list of values is possible. Boolean values are limited to select between one of two constant values, or their variation. The *Add Batch* function automatically generates all spectrum of scenarios given the defined value ranges, while the *Add Single* function enables the generation of single scenarios.

C. Design Stage

The most significant contribution of our MAP++ framework is at the design phase. The visualization of maps leads to a better understanding of design choices. The designer can use our tools to test tentative solutions and their preliminary performance. In the exemplary case of energy monitoring, the visualization may show the regions susceptible to partitioning, their shape and expected time of occurrence. This knowledge results in the development of valuable countermeasures for design or deployment. For example, the threatened regions of the network can be supplied with higher energy reserves, e.g., through denser deployment of nodes. Considering also the visualization of the physical phenomena such as temperature allows for an interactive design of algorithms. Such a visualization allows for a coarse-grained understanding of both problems and developed solutions. For example visualizing the energy distribution of the network along the network topology allows the identification of energy holes and some of their properties.

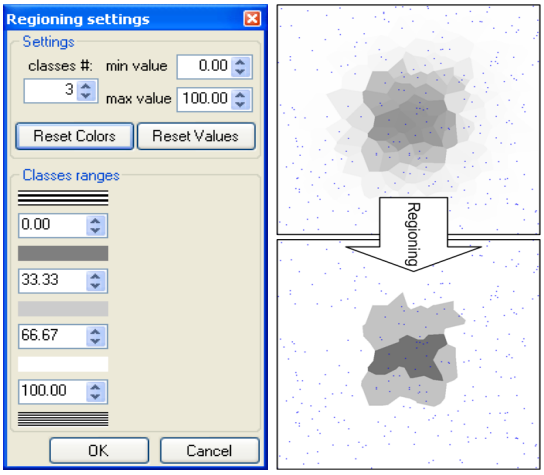
A static snapshot does not provide enough insights into the dynamics of the network. Therefore, the MAP++ visualization tools support smooth scrolling through the trace data along the time axis. This procedure unveils the causality of the events during the lifetime of the network. An example could be the investigation of the adaptability of a routing protocol to sensor nodes failures. For this purpose the designer may create a connectivity map, which visualizes the hop distance of each node to the sink. The visualization of this map (small differences in grey scale values show neighboring sensor nodes) reveals when and how the routing adjusts to changes in the network. Observing the network at the same time in different perspective (e.g., energy and connectivity map) points the cause of rerouting.

The impact of varying parameters becomes visible when comparing the different scenarios at same time indexes. Visualization allows loading several traces at

once and switching between their views, remaining at the same time index.

Testing the solutions and countermeasures for the identified problems requires the generation of additional scenarios. These scenarios should not only vary the simulator's parameters but also the spatial distribution of initial conditions. Manual text edition of the topology is a demanding and error prone task. Additionally, taking the spatial correlation of the initial values into consideration makes the process of scenario definition even more complex. Using the graphical interface and network visualization at design time, it is easy to identify and select neighboring sensor nodes and assign to them attribute values in collective manner. In case of energy monitoring the user can easily create the desired energy distribution simulating higher energy densities in more threatened regions.

MAP++ also allows for map-based tracing. Such a trace gives the set of regions constituting the map for a certain time interval and lists the regions splitting and merging operations during that time. In order to set region information we implemented a *regioning* technique. Our Regioning algorithm logically groups neighboring sensor nodes belonging to the same value class (defined values range). Fig. 4(a) shows the value classes definition form. The designer can set the number of the classes and the range of the values defining the class membership. For visualization purposes, every class is assigned a color, which is used to represent it on the map (Fig. 4(b)).



(a) Regioning Settings (b) Regioning

Figure 4. Regioning in MAP++

D. Debugging and Performance Evaluation Stage

An important activity in the implementation of an algorithm is debugging. The map perspective simplifies the task of localization of a spatially correlated errors.

For example the energy distribution map can be used to identify routing algorithm implementation errors. An energy map showing the occurrence of energy holes at unexpected locations, may indicate such routing implementation errors. The connectivity map offers additional information about the nature of the problem, e.g., when showing irregular hop distances to the sink.

The data aggregation errors can be identified by comparison of the ideal map with the one created by the aggregation algorithm. Comparison is directly supported by MAP++ and is achieved by creating the differential map of both views. The values of two selected attributes of each sensor node (e.g., real and aggregated value) are subtracted from each other and the resulting value is used for coloring the area occupied by the corresponding sensor node.

Besides visualization, the trace data can be evaluated using the MAP++ database interface. The query tool searches the database for a set of changing parameters and creates a list of their distinct values. The user may choose the range of the values from the list in order to simplify the creation of a database query. The selected values are automatically encoded as a query string (*Prepare Query* button, Fig. 3), so even users with limited knowledge of SQL can use the database interface.

As mentioned before, the query tool is also integrated with the visualization. To find the snapshot of interest, the user may use the visualization for navigating to the desired time index. Additionally, using the visualization interface it is possible to select only the nodes that should be addressed in the query. The query tool automatically generates the query that uses current time index and references only the selected nodes.

E. Usage Guidelines

For further illustration of the functionality offered by MAP++, we present here a few guidelines for using the framework. The MAP++ framework is mainly intended as a support of WSN simulation. In particular, it is most suitable for algorithms/applications that directly or indirectly exploit the space correlation of the events/measurment in the network. Its main strength is a straight-forward visualization of the distributions in space of attributes (e.g., energy, temperature, route length, etc.) among sensor nodes in form of a map. At the simulation planning stage the visualization can be used to create simulation scenarios by setting spatially correlated initial conditions (initial energy/temperture distribution). At the post execution stage the visualization allows for comprehensive evaluation of the spatial distribution of selected attributes in the network. The visual operations on different perspectives of the network uncover the correlations between the attributes. These abstract view is supplemented by the use of database engine for processing of data. That approach allows easy extraction of detailed information from the high level abstraction.

VI. MAP++ Evaluation

We now demonstrate the powerful utilities of the developed MAP++ framework through two representative case studies.

- Case Study 1 was chosen to demonstrate the advantages that the map paradigm brings for simulation of

common WSN applications. It shows how the framework supports an accurate but efficient evaluation of the eScan - data collection algorithm.
- Case Study 2 highlights the capabilities of the MAP++ framework for comparison of algorithms.

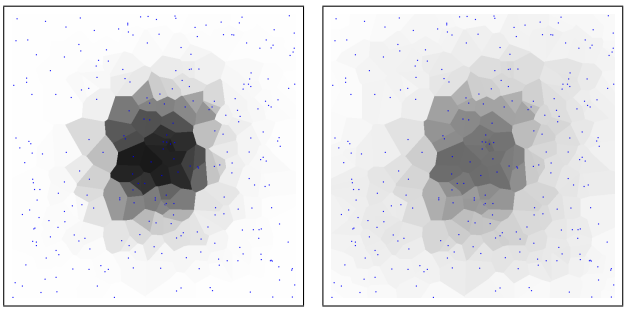
The results obtained from the case studies show MAP++ helps identify key algorithm properties that other existing simulations tools were unable to provide. Finally, we evaluate the framework with respect to its impact on the scalability of OMNeT++.

A. Case Study 1 - Algorithm Evaluation

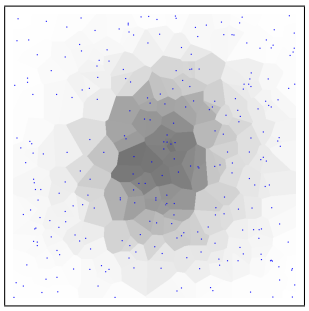
Using the example of *eScan* [5] we demonstrate the utility of the MAP++ framework. To this end, we first briefly describe eScan and then use the MAP++ visualizations to select the energy consumption model. Finally, we investigate the performance of eScan and show results that the original paper could not obtain.

1) Algorithm Description: The eScan [5] energy map construction approach is based on polygon aggregation. The sink disseminates the interest/query for a map to all network nodes. The query is disseminated using flooding to create a spanning tree rooted at the sink. This tree is used to aggregate the attribute values while being reported. A leaf node sends its raw value to its parent node. An internal node (parent) gathers the input of all its children, aggregates it with its value and forwards the aggregate to its parent node and so on. The aggregation consists of grouping sensor readings that meet a specified criteria (being geographically adjacent and in the same value range). The degree of aggregation is defined by the *tolerance* parameter *T*, corresponding to the percentage difference between potential attribute values to be merged. The outcome of an aggregation operation is a list of (spatial) regions. A region is a polygon that is defined by the line spanning its border sensor nodes. At the sink the aggregation results in a complete map. Sensor nodes immediately reply with their current values on a query, and later only do updates (*update interval* parameter defined as the percentage value change, after which the sensor nodes transmit an update).

2) Evaluation Settings: eScan has been evaluated for three hotspot energy distribution models: Exponential, pareto and normal. We easily visualize the resulting energy maps using MAP++ in Figures 5(a), (b), (c) respectively. In hotspot model, each sensor node n performs sensing with a probability $p = f(d)$, where d is Euclidean distance of n to its closest hotspot epicenter. Each sensing activity consumes a fixed amount of energy. For simplification, we arbitrarily concentrate only on one of the hotspot distribution models, namely the exponential distribution. To closely recreate the eScan original scenario, as derived from its source code, we consider a network of 270 sensor nodes spread uniformly over the square area of size 30m x 30m. Sensor nodes have a communication range of 3m. The hotspot epicenter is located in the geographical center of the network area (Fig. 5).



(a) Exponential Hotspot (b) Pareto Hotspot



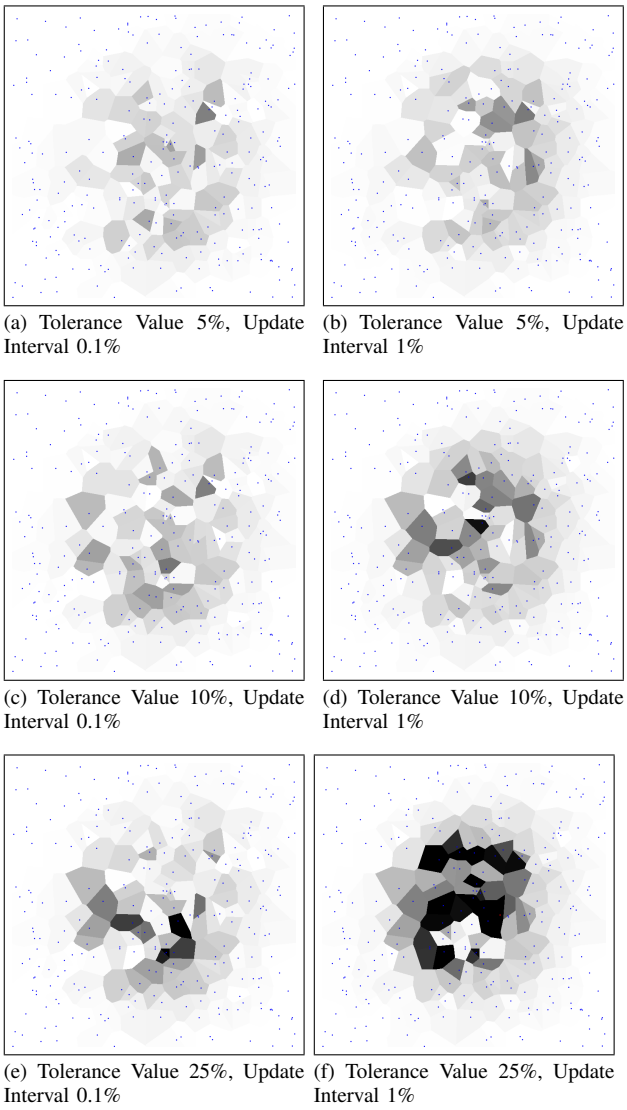
(c) Normal Hotspot

Figure 5. Energy Map for Different Hotspot Energy Distribution Models

Using the MAP++ batch scenario generator we vary the values of the tolerance and update interval parameters. We generate a total of 6 scenarios for tolerance parameter values of 5%, 10% and 25% and update interval parameter values of 0.1% and 1%. We use a relative differential map. This is a modified version of the differential map discussed earlier in Section V as the best illustration of the accuracy of the eScan algorithm. It is defined as a map of percentage difference between measured and actual energy values. To visually represent the map we define the following grey shading schema. To the lowest obtained value 0 (no relative error) we assign the white color. To the maximum, defined as the maximum discrepancy for all scenarios (in this case 30% relative error) we assign the black color.

3) Evaluation Results: The comparison of Fig. 6(a) and Fig. 6(b) shows no significant difference regarding the accuracy of the algorithm as well the localization of error concentration. The hotspot occupied area (hotspot extent we simulated in first step in Fig. 5(a)) is filled only with light shades of grey. Therefore, the use of higher update interval would be advisable to reduce the number of messages transmitted. The results presented in Fig. 6(c) show that despite a higher tolerance value, the accuracy of algorithm is still acceptable. The concentration of grey shades is comparable with the two previous snapshots of Fig. 5(a), 5(b). Fig. 6(d) shows that the tolerance is set too high to compensate the inaccuracy introduced by a higher update level. Significant differences between both sub-figures are manifested by the occurrence of very dark spots in Fig. 6(d). Fig. 6(e) shows that at 25% tolerance value, the eScan map accuracy suffers significantly. This performance degradation is amplified by an increase of

update interval value (Fig. 6(f)).



(a) Tolerance Value 5%, Update Interval 0.1%

(b) Tolerance Value 5%, Update Interval 1%

(c) Tolerance Value 10%, Update Interval 0.1%

(d) Tolerance Value 10%, Update Interval 1%

(e) Tolerance Value 25%, Update Interval 0.1%

(f) Tolerance Value 25%, Update Interval 1%

Figure 6. eScan: Accuracy for Different Simulation Scenarios

When analyzing the visualized results the important added value of our approach becomes evident. In existing approaches, the accuracy is averaged for the entire network. In case of the map visualization the regions that contribute mostly to the error can be easily localized. In case of eScan the conclusion is obvious that the epicenter of hotspot shows highest dynamics of the changes and consequently highest relative and absolute errors.

B. Case Study 2 - Algorithms Comparison

In this case study, we show how the MAP++ framework can be utilized to compare two algorithms to construct maps in WSN, i.e., the eScan algorithm from Case Study 1 and the Isoline algorithm [3]. Subsequently, we describe the evaluation methodology and comparison results.

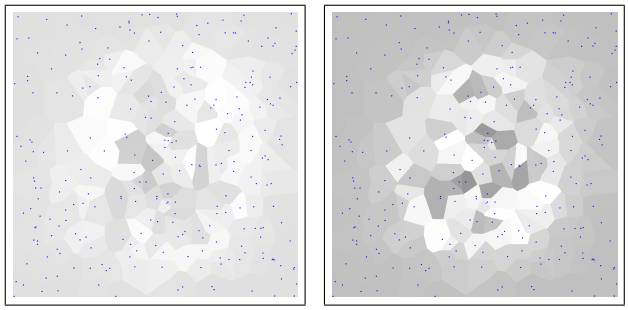
1) The Isolines Algorithm: The Isolines algorithm [3] represents an aggregation technique designed for map-based collection and visualization of spatially correlated data. It uses the notions of isolines. Isolines are lines connecting the points of equal measurement value to present specified data as a map. Nodes located between two or more neighboring isolines are nodes that show sensor values within the same class defined as ranges (e.g., (0-10], (10-20]). The Isolines algorithm consists of two main phases: (1) Sensor nodes detect the isolines through inter-sensor-node beaconing, and (2) sensor nodes that have detected an isoline send specified reports to the sink, which is responsible for deriving the isolines from the submitted reports. To detect the isolines, each node compares its value with the one of its neighbors. If the values belong to different value classes, then an isoline is detected. In order to construct the isolines map at the sink, sensor nodes that detect an isoline passing by, report their sensor value/class and their location to the sink. In order to reduce redundant reports and save node energy and communication bandwidth, from two neighboring nodes that have detected that an isoline is between them, only that sensor node closer to the sink sends its report. The isolines divide the network area into regions, and these regions are shaded using grey scales.

The Isolines technique requires very limited processing of data inside the network as opposed to the eScan approach where nodes need to derive new polygon at each aggregation step. Also the size of transmitted reports by the Isolines approach is lower than in eScan. In the Isolines approach the reporting senor nodes send only their value and those of their neighbors. The eScan approach also requires including the description of the aggregated polygon. Moreover, the efficiency of aggregation for eScan strongly depends on the routing spanning tree. The Isolines approach accuracy is independent of network topology. On this basis, we now qualitatively compare the accuracy of data collection performed by eScan and Isolines.

2) Comparative Study and Evaluation Settings: The collection of data in the Isolines technique is simple, however, the construction of the isolines at the sink is a complicated geometric implementation. As MAP++ allows multi-level implementation, this helps compare the selected features of the algorithms. As the described Isolines concept is analogous to our Regioning technique (Section V-C), thus instead of implementing the Isolines algorithm, we use the Regioning technique to derive the isolines and split the network area in regions. In order to directly compare the accuracy of the Isolines and eScan approaches, we selected the simulation parameters so as to produce comparable results. We use the obtained raw data from Case Study 1 and run the Regioning algorithm with value ranges of 5% (e.g.: (0%-5%], (5%-10%]), 10% (e.g.: (0%-10%], (10%-20%]) and 25% (e.g.: (0%-25%], (25%-50%]) of the peak hotspot value. We use value ranges expressed as percentage of the peak hotspot value, instead of absolute values, for easier comparison with the eScan approach. For simulating Isolines we do not consider the second parameter of Case Study 1 Update Interval, as reporting by Isolines is not delayed

and happens promptly after the node changes its region. Therefore, we compare the Isolines results with lower value of Update Interval of eScan. If not otherwise mentioned, the rest of the evaluation settings remains the same as in Case Study 1.

3) Comparison Results: We first derive the relative differential maps showing the distribution of the inaccuracy in the Isolines algorithm (Fig. 7). Contrary to the eScan algorithm, the inaccuracy appears not only in the vicinity of the hotspot but also spreads across the rest of the network area. The figures, when compared to the relative differential map of eScan, also clearly show that the Isolines approach provides significantly lower accuracies for analogous settings. It is especially apparent in case of 10% and 25% value ranges (Fig. 7 (b) and (c)).



(a) 5% Range Values (b) 10% Range Values

(c) 25% Range Values

Figure 7. Isolines: Accuracy for Different Simulation Scenarios

We compare both algorithms by generating the differential map of the corresponding relative differential maps of each algorithm. This simplifies pointing out (a) which algorithm and (b) in which geographic area outperforms the other one. For this purpose, we first subtract the inaccuracy values of eScan from those of Isolines (Fig. 8 (a), (c) and (e)) and shade the regions where the results have positive values, meaning that the Isolines inaccuracy is higher than that of eScan. Next, we subtract the inaccuracy values of Isolines from those of eScan (Fig. 8 (b), (d) and (f)). The white color marks the regions where the other algorithm performs better. The shades of grey show how much lower is the accuracy of the given algorithm. The grey scale is the same for all figures.

Comparing pairwise the figures (Fig. 8 (a) vs (b), (c) vs (d) and (e) vs (f)) confirms that overall the Isolines algorithm provides lower accuracy than the eScan algorithm. For high resolution aggregation of 5% tolerance and range values (Fig. 8 (a) and (b)), the results are similar. The next two pairs show growing discrepancy between the eScan and Isolines algorithms. The higher the tolerance and range values, the more regions show higher inaccuracy for the Isolines algorithm. The higher inaccuracies of eScan concentrate close to the epicenter of the hotspot. We explain this noticeable accuracy difference by the fact that the eScan aggregation depends also on the spanning tree of routing protocol. This dependency reduces the scope of aggregation regions, forcing the transmission of data, which under given accuracy may be deemed redundant. Isolines better utilizes the aggregation potential of neighboring regions, which comes at the cost of higher inaccuracies. The few spots on the differential map show better performance of the Isolines approach. This can be explained by the fact, that some of the sensor nodes lie at locations, where measurements are close to the middle value of the defined value range.

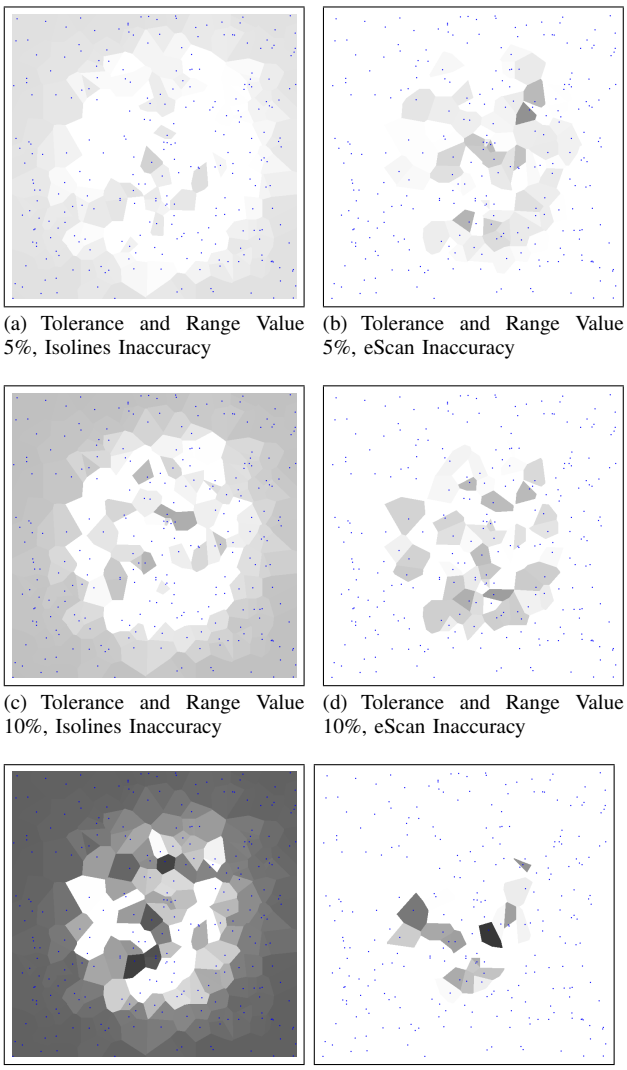
These obtained results contradict the comparison results stated by the authors of the Isolines technique. There are several issues leading to this situation. The first is the applied metric. We derive the differential relative map as an inaccuracy percentage of the measured value. The Isolines approach computes the average distance between corresponding points obtained from no aggregation, isoline, and polygon aggregation as a metric. We measure accuracy per node. The Isolines approach measures accuracy for reconstructing the shape of the isoline. The metrics are fundamentally different and show different qualities of the algorithms. The second issue leading to contradicting conclusions is the evaluation scenario chosen. We simulate the phenomenon with intensity values between 0 and 100 units at peak value, generating as result several layers of isolines. The Isolines approach evaluates the case of a single isoline separating two regions. In this case, even the wide but properly tuned value range still allows proper reconstruction of a single isoline. The scenario used in our cased study requires more adaptability from the algorithm, which, without prior knowledge, the Isolines approach lacks.

C. Performance Evaluation of MAP++

With respect to the performance evaluation of MAP++ the most important factor is the additional simulation time overhead generated by the Trace Module. This determines the usability and the overall scalability of MAP++ and its OMNeT++ environment. We use as a metric the ratio between simulation time with and without MAP++ Trace Module. We consider the impact of the number of sensor nodes and the snapshots interval period.

Fig. 9(a) presents the relative execution time overhead for varying number of sensor nodes (between 250 and 750) keeping the snapshot period at 200s of simulation time. Overall, the time overhead remains fairly constant and at approximately 5%.

Fig. 9(b) depicts the impact of changing snapshot period from 50s to 800s of simulated time, for a fixed



(a) Tolerance and Range Value 5%, Isolines Inaccuracy

(b) Tolerance and Range Value 5%, eScan Inaccuracy

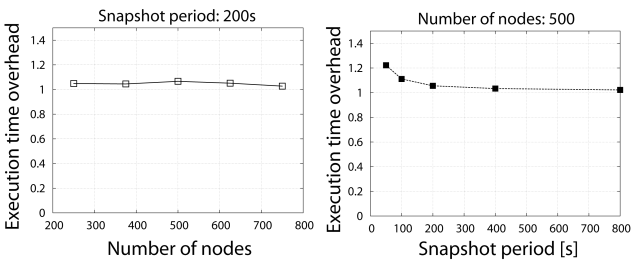
(c) Tolerance and Range Value 10%, Isolines Inaccuracy

(d) Tolerance and Range Value 10%, eScan Inaccuracy

(e) Tolerance and Range Value 25%, Isolines Inaccuracy

(f) Tolerance and Range Value 25%, eScan Inaccuracy

Figure 8. eScan and Isolines Inaccuracy Comparison



(a) Influence of Nodes Number

(b) Influence of Sampling Period

Figure 9. MAP++ Performance

number of sensor nodes (500) on the execution time. The choice of the snapshot period is strongly dependent on the evaluated algorithm. The range we have chosen is based on the properties of the evaluated eScan algorithm and adjusted to its dynamics. In case of very short periods values (50s), the snapshot activities dominate the execution time (snapshots are taken more often than the occurrence of simulation events related to the eScan) resulting in a higher time overhead.

The results clearly show that impact of the overhead on the performance of simulator is limited. Moreover, it is possible to tune this impact by properly selecting either size of the simulation (number of simulated sensor nodes) or by adjusting the period at which snapshots are taken.

Another issue for the performance of MAP++ is the file size of generated trace data. MAP++ generates files of acceptable sizes. For example, a simulation consisting of 500 snapshots of 750 sensor nodes, corresponds to a 20 MB trace file.

VII. Conclusions

In the field of the WSN research, maps are an intuitive abstraction of the network. They expose the spatial nature of the network attributes and allow addressing regions instead of single sensor nodes. Layering the set of the maps discloses the dependencies existing in the network. The MAP++ framework constitutes a comprehensive set of tools providing considerable support for the map abstraction. It supports all the main steps of network design efforts starting with the user definition of the sensor nodes, through generation of the topologies and scenarios based on variation of simulation parameters. MAP++ also supports the visualization of the results both in spatial and time domain along with interactive comparison of the maps and SQL based results analysis.

VIII. Acknowledgments

We thank the authors of the eScan algorithm for providing us the source code of their implementation.

References

[1] A. H. Robinson, J. L. Morrison, P. C. Muehrcke, A. J. Kimerling, S. C. Guptill, *Elements of Cartography*. New York: John Wiley & Sons, 1995, 6th Edition.

[2] Y. Liu and M. Li, "Iso-Map: Energy-Efficient Contour Mapping in Wireless Sensor Networks," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2007, p. 36.

[3] I. Solis and K. Obraczka, "Isolines: energy-efficient mapping in sensor networks," in *IEEE Symposium on Computers and Communications (ISCC)*, 2005, pp. 379–385.

[4] X. Meng, T. Nandagopalz, L. Li, S.Lu, "Contour maps: Monitoring and diagnosis in sensor networks," *Computer Networks*, vol. 50, no. 15, pp. 2820–2838, 2006.

[5] Y. Zhao, R. Govindan, D. Estrin, "Residual energy scan for monitoring sensor networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, 2002, pp. 356–362.

[6] W. Xue, Q. Luo, L. Chen, Y. Liu, "Contour Map Matching For Event Detection in Sensor Networks," in *ACM Special Interest Group on Management Of Data (SIGMOD)*, 2006, pp. 145–156.

[7] M. Li, Y. Liu, L. Chen, "Non-Threshold based Event Detection for 3D Environment Monitoring in Sensor Networks," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2007, p. 9.

[8] Md. V. Machado, O. Goussevskaia, R. Mini, C. G. Rezende, A. Loureiro, G. Mateus, J. Nogueira, "Data dissemination in autonomic wireless sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 12, pp. 2305–2319, 2005.

[9] O. Goussevskaia, Md. V. Machado, R. Mini, A. Loureiro, G. Mateus, J. Nogueira, "Data dissemination based on the energy map," *IEEE Communications Magazine*, vol. 43, no. 7, pp. 134–143, 2005.

[10] K. Ren, K. Zeng, W. Lou, "Secure and fault-tolerant event boundary detection in wireless sensor networks," *IEEE Transactions on Wireless Communications*, vol. 7, no. 1, pp. 354–363, 2008.

[11] R. Sarkar, X. Zhu, J. Gao, L. Guibas, J. Mitchell, "Iso-Contour Queries and Gradient Routing with Guaranteed Delivery in Sensor Networks," in *IEEE Conference on Computer Communications (INFOCOM)*, 2008, pp. 960–967.

[12] A. Khelil, F. K. Shaikh, A. Ali, N. Suri, "gMAP: an efficient construction of global maps for mobility- assisted wireless sensor networks," in *Conference on Wireless On demand Network Systems and Services (WONS)*, 2009, pp. 189–196.

[13] A. Khelil, F. K. Shaikh, B. Ayari, N. Suri, "MWM: A Map-based World Model for Wireless Sensor Networks," in *Autonomics*, 2008.

[14] "OMNeT++ Community Site," http://www.omnetpp.org/.

[15] P. Szczytowski, A. Khelil, N. Suri, "POSTER: MAP++: Support for Map-Based WSN Modeling and Design with OMNeT++," in *OMNeT++ Workshop*, 2009.

[16] "wSCoop: WIRELESS SENSOR COOPERATION," http://www.deeds.informatik.tu-darmstadt.de/dewsnet/.

[17] P. Szczytowski, A. Khelil, N. Suri, "Map-Based Modeling and Design of Wireless Sensor Networks with OMNeT++," in *SPECTS*, 2009, pp. 162–169.

[18] P. Levis, N. Lee, M. Welsh, D. Culler, "Tossim: accurate and scalable simulation of entire tinyos applications," in *ACM Conference on Embedded Networked Sensor Systems (SenSys)*, 2003, pp. 126–137.

[19] "TinyOS," http://www.tinyos.net/.

[20] S. McCanne and S. Floyd, "NS Network Simulator," http://www.isi.edu/nsnam/ns/.

[21] "Mannasim Framework," http://www.mannasim.dcc.ufmg.br/.

[22] Y. Xue, H. S. Lee, M. Yang, P. Kumarawadu, H. H. Ghenniwa, W. Shen, "Performance evaluation of ns-2 simulator for wireless sensor networks," in *Canadian Conference on Electrical and Computer Engineering*, April 2007, pp. 1372–1375.

[23] J. Lessmann, T. Heimfarth, P. Janacik, "ShoX: An Easy to Use Simulation Platform for Wireless Networks," in *IEEE International Conference on Computer Modelling and Simulation (UKSIM)*, 2008, pp. 410–415.

[24] D. Estrin, M. Handley, J. Heidemann, S. McCanne, Y. Xu, H. Yu, "Network visualization with nam, the vint network animator," *Computer*, vol. 33, no. 11, pp. 63–68, 2000.

[25] T. Krop, M. Bredel, M. Hollick, R. Steinmetz, "Jist/mobnet: combined simulation, emulation, and real-world testbed for ad hoc networks," in *ACM international workshop on Wireless network testbeds, experimental evaluation and characterization (WinTECH)*, 2007, pp. 27–34.

[26] G. Chen,J. Branch, M. J. Pflug, L. Zhu, B. K. Szymanski, "SENSE: A Sensor Network Simulator," *Advances in Pervasive Computing and Networking*, pp. 249–267, 2004.

[27] S. Ahmed, M. Bilal, U. Farooq, N. Fazl-e-Hadi, "Performance Analysis of various routing strategies in Mobile Ad hoc Network using QualNet simulator," in *International Conference on Emerging Technologies (ICET)*, 2007, pp. 62–67.

[28] A. Sobeih, W. Chen, J. C. Hou, L. Kung, N. Li, H. Lim, H. Tyan, H. Zhang, "J-Sim: a simulation and emulation environment for wireless sensor networks," *IEEE Wireless Communications*, vol. 13, no. 4, pp. 104–119, 2006.

[29] "Mobility Framework for OMNeT++," http://mobility-fw.sourceforge.net/.

[30] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P.T. Klein Haneveld, T.E.V. Parker, O.W. Visser, H.S. Lichte, S. Valentin, "Simulating Wireless and Mobile Networks in OMNeT++ The MiXiM Vision," in *OMNeT++ Workshop*, 2008.

[31] H. N. Pham, D. Pediaditakis, A. Boulis, "From Simulation to Real Deployments in WSN and Back," in *IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, 2007, pp. 1–6.

[32] "EYES WSN Simulation Framework," http://wwwes.cs.utwente.nl/ewsnsim/.

[33] "NesCT: A language translator," http://nesct.sourceforge.net/.

[34] T. Dreibholz and E. P. Rathgeb, "A Powerful Tool-Chain for Setup, Distributed Processing, Analysis and Debugging of OMNeT++ Simulations," in *OMNeT++ Workshop*, 2008.

[35] J. Lessmann and T. Heimfarth, "Flexible Offline-Visualization for Mobile Wireless Networks," in *IEEE International Conference on Computer Modelling and Simulation (UKSIM)*, 2008, pp. 404–409.

[36] vast, "Space Time Toolkit," http://vast.uah.edu/.

[37] F. Aurenhammer, "Voronoi diagrams - a survey of a fundamental geometric data structure," *ACM Comput. Surv.*, vol. 23, no. 3, pp. 345–405, 1991.

[38] X. Wu, G. Chen, S. Das, "On the Energy Hole Problem of Nonuniform Node Distribution in Wireless Sensor Networks," in *IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2006, pp. 180–187.

[39] J. Li, P. Mohapatra, "An analytical model for the energy hole problem in many-to-one sensor networks," in *IEEE Vehicular Technology Conference (VTC)*, 2005, pp. 2721–2725.

Piotr Szczytowski received his MSc in Computer Science in 2005 from Silesian University of Technology, Gliwice, Poland. Since 2008 he is pursuing his PhD in Germany. Currently, he is a Research Team Member at Technische Universitt Darmstadt, Germany, in the Dependable, Embedded Systems & Software Group. His main research interests include the areas of dependable wireless sensor networks and mobile ad hoc networks.

Abdelmajid Khelil received his MSc in Electrical Engineering in 2000 and his PhD in Computer Science in 2007 from the University of Stuttgart, Germany. Currently he is a Research Team Leader at Darmstadt University of Technology, Germany, in the Dependable, Embedded Systems & Software Group. His main research interests include the areas of dependable wireless sensor networks and mobile ad hoc networks, IT-security and critical infrastructure protection. He leads the research activities in several national, European and transatlantic research projects. He is a member of SCS, ACS, GI and IEEE.

Neeraj Suri holds the TUD Chair Professorship in Dependable Systems and Software at TU Darmstadt, Germany. His research interests span the design, analysis and assessment of trustworthy software and systems. His research activities have garnered extensive support from the EC, German DFG, NSF, DARPA, AFOSR, ONR, Microsoft, Hitachi, IBM, etc. He is a recipient of the NSF CAREER Award, Microsoft and IBM Faculty Awards.