Map-Based Modeling and Design of Wireless Sensor Networks with OMNeT++

Piotr Szczytowski, Abdelmajid Khelil, Neeraj Suri
Technische Universität Darmstadt, DEEDS Group
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 to test and validate approaches, simulation environments need to be able to support the various WSN design schemes. Though the 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, that is, modeling, design, implementation and performance evaluation. A case study for a map construction algorithm shows the utility of our tool chain to enhance common statistic-based trace analysis.**

***Index Terms*—Wireless Sensor Networks, Modeling and Visualization, Map-Based Design, Performance Evaluation**

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 local battery energy source and wireless communication provide high flexibility through self* capabilities. For instance self-organization allows the sensor nodes to be randomly deployed by dropping them aerially over the target area. Therefore, the scope of WSN applications is growing very rapidly. Unfortunately, energy constrained lifetime with the main drain incurred by wireless communication is also the cause of the most serious limitations regarding WSNs.

Sensor nodes inherent redundancy and spatial nature of the monitored physical phenomena results in spatially correlated sensor readings and indirectly in network properties, like 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 map construction groups spatially correlated sensor nodes with similar attribute's 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.

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

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], we propose that maps are the natural step towards a holistic *Map-based World Model* and *Map-based WSN design*.

Our Contributions

In most of network simulators, both the design of topologies as well as storing and representation of the trace data have textual format. Expression of the spatial dependencies in that form is both complex and limited. WSN 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 which create topologies following the simulator syntax. Unfortunately, techniques to verify and validate the created topology are limited. A direct and effective solution is visual validation.

The established OMNeT++ [13] 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 other 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 network and the physical world. The lack of support for visualization is even more clear in case of trace data. Currently, trace data is collected in a text file used for statistical analysis. Usually, analysis is done at sensor node or event level. This is tedious work and does not follow the region abstraction level as discussed earlier. Visualization rendered as a map is natural approach for the discovery of spatial dependencies. Of course the map representation cannot fully replace the traditional statistical approaches, which are required for understanding the details of investigated protocols. However, it offers considerable support for high-level designing and debugging. The processing of data across series of simulations requires 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 relational database and querying using database engine.

In this paper, we extend our preliminary work [14] and provide a map-based MAP++ framework [15] that delivers following contributions, which we highlight through a case study:

- Extended topology generation, including setting of spatially correlated initial conditions and batch script generation for simplifying modeling and scenarios generation
- Trace data visualization support that identifies spatial dependencies for design and debugging
- Export to and processing of trace data with support of relational databases for performance evaluation

The remainder of the paper is structured as follows. Section II describes the assumed system model. Related work is presented in Section III. Section IV details the architecture and implementation of MAP++. In Section V illustrations of application scenarios are presented. The framework performance and usability is analyzed in Section VI and the paper is concluded in Section VII.

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. Sensor nodes measure the physical signal at given sampling rate (e.g. temperature, humidity, as well as its own battery status). There also exists one or more base stations (called sinks), which are designated resource-rich nodes serving as a gateways between the WSN and its operator 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 WSN research resulted in development of a variety of simulation environments. TOSSIM [16] being a simulator for the common TinyOS platform [17] serves as a very good example of such environment. Its attractive feature is that code developed for the simulator can be directly deployed on the physical 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, they also put constraint on network scale, rendering the simulator inapplicable for large scale simulations.

Also some of existing network simulators are becoming adapted to support simulation of WSN. One of the examples is the established NS-2 simulator [18]. The MannaSim Framework [19] provides such an extension by delivering the topography generation tool, modules simulating an antenna, and radio propagation models. However, till now the capabilities of NS-2 simulator in the area of WSN are very limited due to its limited scalability [20].

Projects like [16], [21]–[26] deliver rich libraries of components and algorithm templates leveraging different aspects of WSN and some of them offer various forms of visualization. All of them are targeted for displaying the messages flows and debugging rather than rendering the attributes distribution in regards to map paradigm.

OMNeT++ [13] 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 generic and flexible open-architecture, provides extensive GUI interface and is platform independent. These characteristics make it especially suitable for highly efficient network simulations. Owing to its modularity and as consequence extensibility it receives a growing recognition in the WSN community resulting in the implementation of mobile and sensor frameworks for this simulator. Example of such effort is the Mobility Framework for OMNeT++ [27], which is intended to support wireless and mobile simulations within OMNeT++. It provides features like sensor node mobility, dynamic connection management and a wireless channel model. A similar approach is presented in MiXiM [28], 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 MAC protocols and localization algorithms. Also [29] builds up on top of OMNeT++ delivering an advanced radio/channel model allowing multiple transmission power levels, a physical process model, and a resource monitoring MAC protocol.

Authors of *EYES WSN Simulation Framework* [30] and *NesCT* [31] projects investigated the possibility to use TOSSIM simulations in OMNeT++ and provide necessary extension for handling wireless networks. The simulation profits from rich library of components written for TinyOS, increased efficiency, and extensive GUI provided by OMNeT++. EYES also uses maps, but only as mean of defining the failing probabilities of deployed sensor nodes or modeling the radio propagation by marking the obstacles. However, they are not the attribute distribution maps. [32] presents the tool-chain for providing the parametrization, distributed execution and result-postprocessing and debugging. Similar to our approach, [33] provides offline visualization of traces. The difference is that [33] works on the node-level basis, not utilizing the fact of spatial correlation of sensor nodes attributes. The tool is only limited for offline replaying of simulations, and provides no support for scenario generation and evaluation.

The multitude of mentioned works regarding WSN confirms the rising interest in developing the simulation environment for wireless networks. Unfortunately, no project within the WSN community provides a holistic support for map design and modeling in simulation environments. Approaches in other communities like Space Time Toolkit [34] provide advanced capabilities for integrating spatially and temporally-disparate data within 2D or 3D display domain. Unfortunately, lack of access to source code renders the 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

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

A. Overview of the MAP++ Architecture

The presented problem and system model drive the MAP++ design. Fig. 1 explains the MAP++ framework architecture and interactions between the framework and the OMNeT++ simulator. Simulation studies usually require a large number of scenarios to evaluate a vast parameters domain. *Scenario Generator* is responsible for generation of the topologies designed by the user and supplemented by simulation scripts for batch execution using varied parameters. The framework requires 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 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 defined scenarios and embedded trace modules produce *Trace*s. Traces are stored in XML format, which makes them suitable for automated parsing and subsequently usable for producing maps. 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* allows further map operations by providing the grouping of the nodes depending on their class membership, where a class is defined as a value range of the selected map attribute. More accurate and statistical evaluation of data is provided through the *SQL Database Interface* module. It allows the import of XML formatted *Trace* data into the relational database and the querying data using database engine. The query creation is simplified by integration of database module with the visualization module, where users may visually create queries.

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

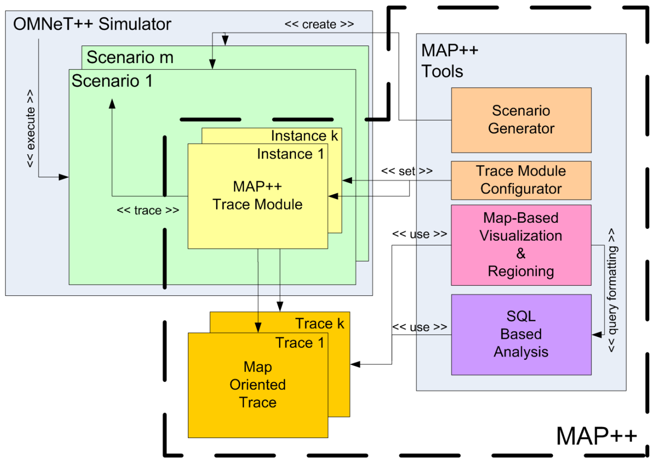


Fig. 1. MAP++ Architecture

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. The modules developer should not be required to integrate the necessary traces into modules source code. The 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, for each of the module classes, a separate instance of the Trace Module can be initialized, allowing parallel tracking of different sets of network perspectives. For example if network contains two different classes of modules each equipped with different sensor (e.g. temperature and humidity sensor nodes) then two instances of Trace Module are need, as each Trace Module may trace only one module class. In case that single module class provides two or more sensors, then only single Trace Module is necessary for tracking.

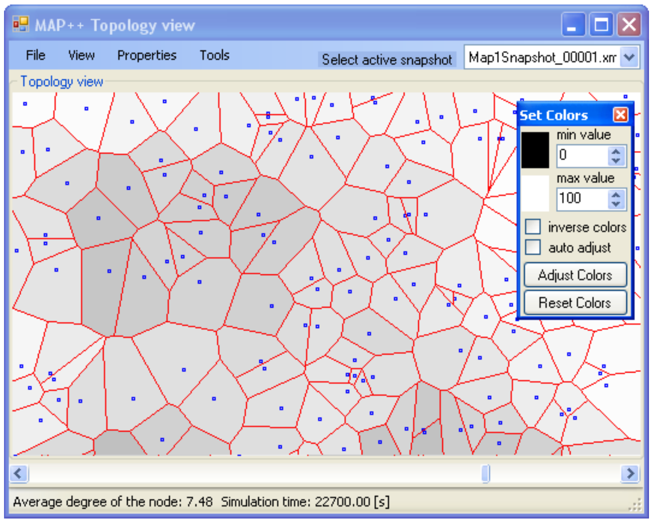
C. The MAP++ Tools

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 defined number of sensor nodes within 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 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 [35] 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 provided using the 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 region.



Select active snapshot Map1Snapshot_00001.xm

min value 0

max value 100

Average degree of the node: 7.48 Simulation time: 22700.00 [s]

Fig. 2. Voronoi-Based Map Visualization

V. MAP++ Benefits and Applications

In this section, we first present main functionalities of the framework at different stages of simulation process. Then we describe how these functionalities can be applied. For this purpose we present the main usage scenarios, which include modeling, design, debugging and performance evaluation. To better illustrate the framework's capabilities we use an OMNeT++ implementation of the eScan algorithm [5]. eScan is an efficient energy map construction algorithm based on polygon aggregation. A detailed case study using this algorithm is described in Section VI-A.

A. Overview

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.

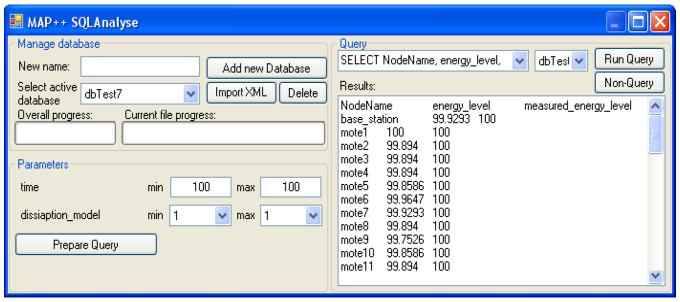
In the modeling stage MAP++ supports the definition of the modules, generation of the topology consisting of defined modules and creation of scenario sets. The definition of a new module consists of the implementation of a new module class and defining set of class attributes. The implementation of attributes on the developers side is limited to updating their value inside the module source code. The topology generation results in an automatized 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 only 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 altered or snapshots apart from being periodical become event triggered. Other possibility of extending the framework functionality at this stage would be the run-time data visualization and real-time 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 two dimensional map, colored 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 like creation of differential maps are supported. Scrolling along time axis displays the progress of simulated phenomenons. 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. Current implementation is provided for Microsoft SQL Server and MySQL open source database.

B. Modeling Stage

We illustrate the modeling capabilities of our tools through the example of energy consumption modeling. As a result



New name:
Select active database: dbTest7
Overall progress:
Current file progress:

time min 100 max 100
dissiaption_model min 1 max 1

SELECT NodeName, energy_level, — dbTest

Results:

```
NodeName          energy_level        measured_energy_level
base_station      99.9293   100
mote1     100       100
mote2     99.894    100
mote3     99.894    100
mote4     99.894    100
mote5     99.8586   100
mote6     99.9647   100
mote7     99.9293   100
mote8     99.894    100
mote9     99.7526   100
mote10    99.8586   100
mote11    99.894    100
```

Fig. 3. Database Interface

of the WSN inherent sensor nodes redundancy, the energy depletion tends to show high spatial correlation. The sensor nodes placed in close proximity of the sink deplete their energy at a much faster rate [36], [37] as, besides their own operations, they 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 reporting frequency has to be adjusted to meet the required accuracy of monitoring, resulting in an increased energy usage ratio.

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 sensor nodes is supported by iteratively adding new module classes to existing topologies.

The topology generation is followed by accurate modeling of the phenomena to be monitored by the WSN. As physical phenomena usually show spatial correlations, the natural approach to model that 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 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

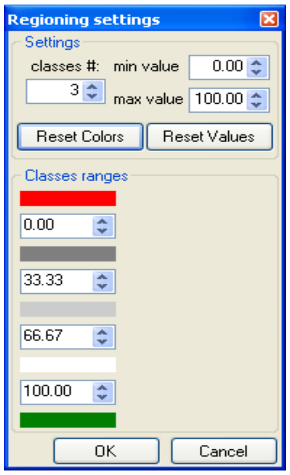
The most significant contribution of our MAP++ framework is at the design phase. The visualization of maps leads to a better understanding of design choices. The designer can use our tools to test tentative solutions and their preliminary performance. In the exemplary case of energy monitoring, the visualization 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 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)).


Regioning settings

Settings

classes #: 3

min value 0.00

max value 100.00

Reset Colors | Reset Values

Classes ranges

🟥

0.00

⬛

33.33

⬜

66.67

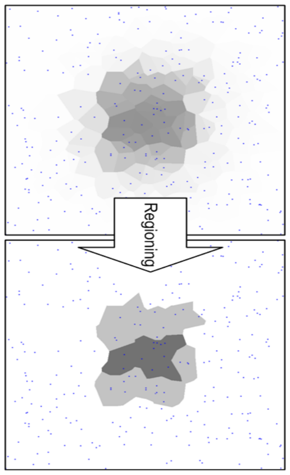
⬜

100.00

🟩

OK | Cancel

(a) Regioning Settings



(b) Regioning

Fig. 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. Energy map showing 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, when showing irregular 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 active 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 database query. The selected values are automatically encoded as a query string (*Prepare Query* button, Fig. 3), so even user with limited knowledge of SQL may use the database interface.

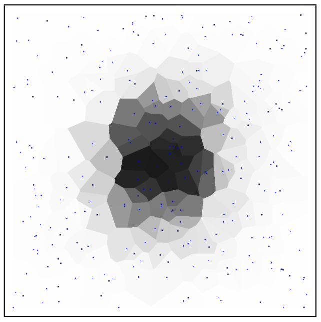
The query tool is also integrated with the visualization. To find the snapshot of interest, the user may use the visualization for navigating to the desired time index. Additionally using the visualization interface it is possible to select only the nodes that should be addressed in the query. The query tool automatically generates the query that uses current time index and references only the selected nodes. An example is the evaluation of the localized accuracy of eScan as we discuss in Section VI-A.

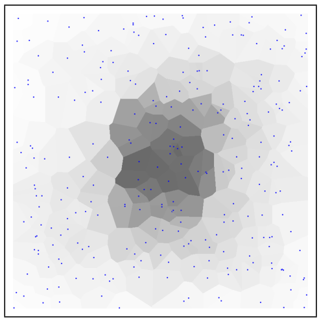
VI. MAP++ Evaluation

First, we demonstrate MAP++ framework powerful utilities through a case study. Next, we evaluate the framework with respect to its impact on the scalability of OMNeT++.

A. *Case Study*

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


(a) Exponential Hotspot


(b) Pareto Hotspot

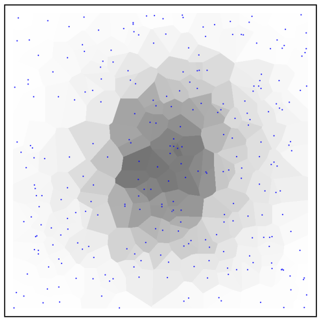
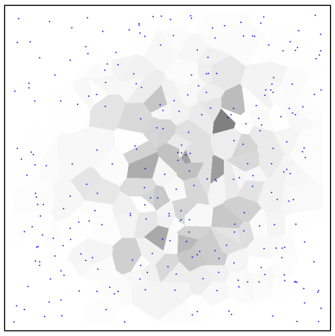
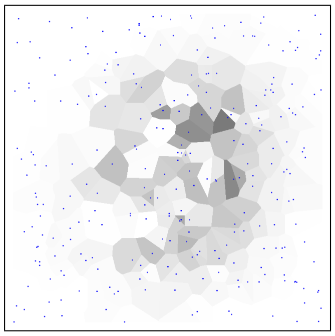
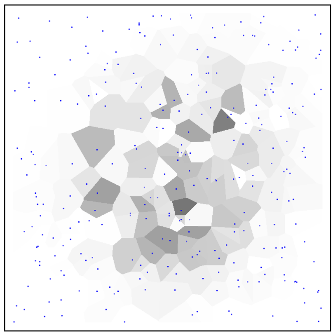

(c) Normal Hotspot

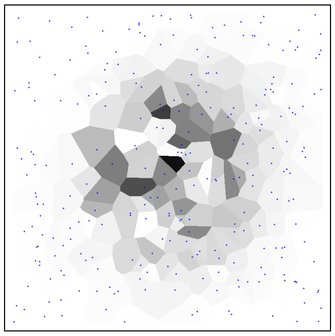
Fig. 5. Energy Map for Different Hotspot Energy Distribution Models

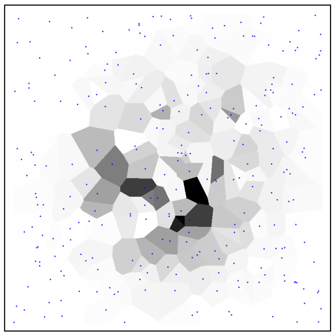
The eScan energy map construction approach is based on polygon aggregation. The sink disseminates the interest/query for a map to all network nodes. The query is disseminated using flooding creating a spanning tree rooted at the sink. This tree is used to aggregate the attribute values while being reported. 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 it to its parent node. The aggregation consists in grouping sensor readings that meet a certain criteria (being geographically adjacent and in the same value range). The degree of aggregation is defined by the *tolerance* parameter *T*, corresponding to the precentral difference between potential attribute values to be merged. The outcome of the aggregation 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 reply with their current values immediately on query and later only with necessary updates (*update interval* parameter defined as the precentral value change, after which the sensor nodes transmit an update).


(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%

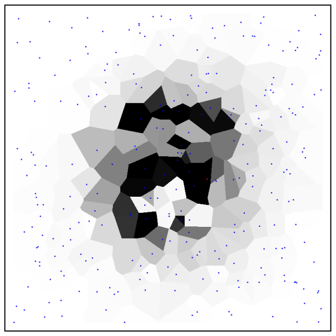

(f) Tolerance Value 25%, Update Interval 1%

Fig. 6. Accuracy for Different Simulation Scenarios)

eScan introduces three hotspot energy distribution models: exponential, pareto and normal, which we easily visualize using MAP++ in Figures 5(a), (b), (c) respectively. In this models each sensor node n has a probability $p = f(d)$, where d is distance to the closest hotspot, that n and its neighboring sensor nodes preform sensing, which consumes fixed amount of energy. For simplification we arbitrarily concentrate only on one of distribution models, namely the exponential distribution. To closely recreate the eScan original scenario, 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 (Fig. 5).

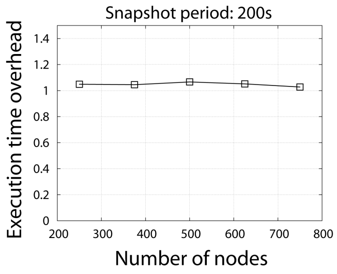
Using the batch scenario generator we vary the values of the tolerance and update interval parameters. We generate in total 6 scenarios for tolerance parameter values of 5%, 10% and 25% and update interval parameter values of 0.1% and 1%. We use relative differential map, which is modified version of the differential map that we discussed earlier in Section V, as the best way of illustrating the accuracy of the eScan algorithm. It is defined as a map of percentage difference between measured and actual value of energy. Coloring the map requires definition of an appropriate color 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. Values between minimal and maximal values are colored in shades of gray. We use the same color schema for each visualization.

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 (which 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 could be advised to reduce number of messages transmitted. The results presented in Fig. 6(c) show that despite higher tolerance value the accuracy of algorithm is acceptable, the concentration of light shades of grey is comparable with two previous snapshots. Fig. 6(d) shows that the tolerance is set too high to compensate the inaccuracy introduced by increased update level and significant differences between both sub-figures are manifested by occurrence of very dark spots in Fig. 6(d). The last set of figures (Fig. 6(e) and Fig. 6(f)) shows that at 25% of tolerance value the eScan map accuracy significantly suffers, and the negative effect is amplified by an increase of update interval value (Fig. 6(f)).

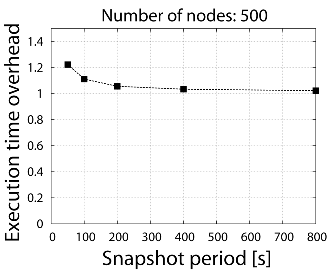
When analyzing the visualized results the important added value of our approach becomes evident. In original results the accuracy is averaged for the entire network. In case of the map visualization the regions that contribute mostly to the error can be easily localized. In case of eScan the conclusion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

B. *Performance Evaluation of MAP++*

With respect to 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 period.



(a) Influence of Nodes Number



(b) Influence of Sampling Period

Fig. 7. MAP++ Performance

Fig. 7(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. We can easily conclude that the time overhead remains fairly constant and equal to approximately 5%.

Fig. 7(b) depicts the impact of changing snapshot period from 50s to 800s of simulated time, for a fixed 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 was based on the properties of evaluated eScan 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 simulation events related to the eScan take place) resulting in a higher time overhead.

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, the maps are the intuitive abstraction of the network. They expose the spatial nature of the network attributes and allow addressing the regions instead of the single sensor nodes. Layering the set of the maps discloses the dependencies existing in the network. MAP++ framework constitutes a comprehensive set of tools providing considerable support for the map abstraction. It supports all main steps of design effort for the network. Beginning with the user definition of the sensor nodes, through generation of the topologies and scenarios based on variation of simulation parameters. Continuing with the visualization of the results both in spatial and time domain, interactive comparison of the maps, and ending with the SQL powered 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 et al. *Elements of Cartography*. John Wiley & Sons, New York, 1995. 6th Edition.

[2] Yunhao Liu and Mo Li. Iso-Map: Energy-Efficient Contour Mapping in Wireless Sensor Networks. In *IEEE ICDCS*, page 36, 2007.

[3] I. Solis and K. Obraczka. Isolines: energy-efficient mapping in sensor networks. In *IEEE ISCC*, pages 379–385, 2005.

[4] X. Meng et al. Contour maps: Monitoring and diagnosis in sensor networks. *Computer Networks*, 50(15):2820–2838, 2006.

[5] Y. Zhao et al. Residual energy scan for monitoring sensor networks. In *IEEE WCNC*, pages 356–362, 2002.

[6] W. Xue et al. Contour Map Matching For Event Detection in Sensor Networks. In *ACM SIGMOD*, pages 145–156, 2006.

[7] M. Li et al. Non-Threshold based Event Detection for 3D Environment Monitoring in Sensor Networks. In *IEEE ICDCS*, page 9, 2007.

[8] Md.V. Machado et al. Data dissemination in autonomic wireless sensor networks. *IEEE Journal on Selected Areas in Communications*, 23(12):2305–2319, 2005.

[9] O. Goussevskaia et al. Data dissemination based on the energy map. *IEEE Communications Magazine*, 43(7):134–143, 2005.

[10] K. Ren et al. Secure and fault-tolerant event boundary detection in wireless sensor networks. *IEEE Transactions on Wireless Communications*, 7(1):354–363, 2008.

[11] R. Sarkar et al. Iso-Contour Queries and Gradient Routing with Guaranteed Delivery in Sensor Networks. In *IEEE INFOCOM*, pages 960–967, 2008.

[12] A. Khelil et al. MWM: A Map-based World Model for Wireless Sensor Networks. In *Autonomics*, 2008.

[13] OMNeT++ Community Site. http://www.omnetpp.org/.

[14] P. Szczytowski et al. POSTER: MAP++: Support for Map-Based WSN Modeling and Design with OMNeT++. In *OMNeT++ Workshop*, 2009.

[15] DEWSNet Dependable Embedded Wireless Sensor Networks. http://www.deeds.informatik.tu-darmstadt.de/dewsnet/.

[16] P. Levis et al. Tossim: accurate and scalable simulation of entire tinyos applications. In *ACM SenSys*, pages 126–137, 2003.

[17] TinyOS. http://www.tinyos.net/.

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

[19] Mannasim Framework. http://www.mannasim.dcc.ufmg.br/.

[20] Yunjiao Xue et al. Performance evaluation of ns-2 simulator for wireless sensor networks. In *Canadian Conference on Electrical and Computer Engineering*, pages 1372–1375, April 2007.

[21] J. Lessmann, T. Heimfarth, P. Janacik. ShoX: An Easy to Use Simulation Platform for Wireless Networks. In *IEEE UKSIM*, pages 410–415, 2008.

[22] D. Estrin et al. Network visualization with nam, the vint network animator. *Computer*, 33(11):63–68, 2000.

[23] T. Krop et al. Jist/mobnet: combined simulation, emulation, and real-world testbed for ad hoc networks. In *WinTECH*, pages 27–34, 2007.

[24] G.Chen et al. SENSE: A Sensor Network Simulator. *Advances in Pervasive Computing and Networking*, pages 249–267, 2004.

[25] S. Ahmed et al. Performance Analysis of various routing strategies in Mobile Ad hoc Network using QualNet simulator. In *ICET*, pages 62–67, 2007.

[26] A. Sobeih et al. J-Sim: a simulation and emulation environment for wireless sensor networks. *Wireless Communications, IEEE*, 13(4):104–119, 2006.

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

[28] A. Köpke et al. Simulating Wireless and Mobile Networks in OMNeT++ The MiXiM Vision. In *OMNeT++ Workshop*, 2008.

[29] H. N. Pham et al. From Simulation to Real Deployments in WSN and Back. In *IEEE WoWMoM*, pages 1–6, 2007.

[30] EYES WSN Simulation Framework. http://wwwes.cs.utwente.nl/ewsnsim/.

[31] NesCT: A language translator. http://nesct.sourceforge.net/.

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

[33] J. Lessmann and T. Heimfarth. Flexible Offline-Visualization for Mobile Wireless Networks. In *IEEE UKSIM*, pages 404–409, 2008.

[34] vast. Space Time Toolkit. http://vast.uah.edu/.

[35] F. Aurenhammer. Voronoi diagrams - a survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23(3):345–405, 1991.

[36] X. Wu, G. Chen, S. Das. On the Energy Hole Problem of Nonuniform Node Distribution in Wireless Sensor Networks. In *IEEE MASS*, pages 180–187, 2006.

[37] J. Li, P. Mohapatra. An analytical model for the energy hole problem in many-to-one sensor networks. In *IEEE Vehicular Technology Conference*, pages 2721–2725, 2005.