Fast Simulation Methods to Predict Wireless Sensor Network Performance*

Miloš Blagojević[1], Majid Nabi[2], Teun Hendriks[1], Twan Basten[1,2], Marc Geilen[2]

[1]Embedded Systems Institute, Eindhoven, the Netherlands
[2]Eindhoven University of Technology, Eindhoven, the Netherlands
milos.blagojevic@esi.nl, m.nabi@tue.nl, teun.hendriks@esi.nl, a.a.basten@tue.nl, m.c.w.geilen@tue.nl

ABSTRACT

With the increasing capabilities of Wireless Sensor Networks (WSN), complexity and expectation of the WSN applications increase as well. In order to make design-space exploration possible, it is necessary to have fast models that provide adequate insight in system behavior. In this paper, we propose a highly abstracted, hierarchical, system-level modeling method for WSN. Based on the model properties, fast simulation techniques can be applied. First, an abstract discrete event simulation based on a Probabilistic Graph Model (PGM) is introduced. Then, a fast Monte Carlo simulation approach is proposed for speeding up the simulation process. This approach combines Stochastic-Variable Graph Models (SVGM), providing a high level of abstraction, with shortest path calculations. As a case study, a temperature mapping application in a gossip-based WSN is used, showing a good accuracy of the model predictions.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: *Wireless communications;* C.4 [**Performance of Systems**]: *Modeling Techniques*

General Terms

Performance, Reliability, Verification.

Keywords

Wireless sensor networks, System modeling, Performance evaluation, Probabilistic abstraction, Simulation, Shortest path, Monte Carlo

1. INTRODUCTION

Wireless Sensor Networks (WSN) are complex systems consisting of spatially distributed autonomous nodes which provide infrastructure for some common application(s). These nodes communicate wirelessly and often are (self) organized into a cooperative network. The nodes integrate some combination of sensing, wireless communication, and signal processing/storage capabilities [15]. Each node incorporates a radio transceiver to provide communication, and a processing unit in order to be able to process data. The sensor nodes are capable of measuring and monitoring the environmental phenomena. Usually a single node can not make good use of the locally collected data. The task of a WSN is to interconnect nodes, collect specified environment data and distribute it over the network. The nodes then collaboratively process gathered information according to predefined application tasks. From the system perspective, the main goal of the WSN is to produce globally meaningful information from locally collected data [2].

Recent advances in both wireless and sensor technology have led to increasing capabilities of WSN and thus increased potential for deployment of WSN for the various applications. WSN are beginning to be deployed at an accelerated pace in many military, industrial and civilian application areas such as healthcare, assisted living facilities, precision agriculture, process monitoring and control [19][3]. Together with increased capabilities, the complexity and expectation of the WSN applications increase as well. WSN are envisioned as a platform for large scale applications which include different tasks such as periodic monitoring, event-based reporting, tracking, prediction, identification and so forth. A WSN application defines the functionality, requirements, and target environment for the designed WSN system [12].

Typically, WSN are expected to operate for several years, to function autonomously without much external control and to be resilient to errors (node failures, interference, sensor errors, …) [15]. On the other hand, WSN are constrained by limited processing capability and memory space, unreliable communication resources, and above all, scarce energy resources [5]. Thus, the primary goal of many WSN architectures is to maximize lifetime of the network by trading off some of the potential system performance (throughput, latency, reliability, …) for a reduction in energy consumption. The tradeoff between lifetime and other performance criteria has to meet application requirements. In order to increase lifetime, most WSN minimize radio communication, since typically the radio expends the most energy in comparison to other components.

Planning and designing WSN requires a deep understanding of the system behavior, both at system and component level. A first step is to define system objectives and requirements. Some of the objectives might be contradictory and thus provide certain tradeoffs. The objective of a designer is to explore the design space and to choose optimal configuration(s). However, for large, non homogenous WSN, the size of the design space is huge, and thus exhaustive exploration is typically not an option. Then, the goal is rather to find a scalable solution that can provide a (near-) optimal configuration in a reasonably short time [10]. Design-Space Exploration (DSE) might be performed by for example genetic algorithms [6, 17], which provide a generally applicably and robust solution. DSE is in need of a tool that can estimate WSN performance in a short time, while providing adequate accuracy. In this work, a fast simulation method with adequate predic-

*This work was partially supported by the Dutch innovation program Point-One, through project ALwEN, grant PNE07007.

PE-WASUN'09, October 28–29, 2009, Tenerife, Canary Islands, Spain.

tion accuracy is proposed as a support tool for DSE. The proposed method is based on the identification of system abstraction layers, and a modeling strategy which faithfully captures the layer's functionality and their interactions.

First, in Section 2, a brief overview of related work is given. In Section 3, temperature mapping is introduced as a case study system and some details are given about the used protocols and technologies. Section 4 introduces the proposed method for performance evaluation. Section 5 presents the system model of the WSN, it defines the metrics and configuration parameters and it proposes a layered system model. In Section 6, based on the proposed model, a novel, fast simulation technique is described. Finally, Section 7 gives some simulation results and analyzes speed and accuracy. Section 8 lists conclusions.

2. RELATED WORK

Predicting the behavior of the WSN is not an easy task. Much work has been done and numerous WSN simulators have been developed, with different goals in mind. However, system-level, multi-domain models of WSN systems are not available in the current state of the art. Instead, researchers need to switch between several different aspect models.

WSN models are often made by modifying general network simulators for WSN systems (ns-2, Omnet++, Opnet). In that case the main focus is on the modeling of wireless communication protocol stacks. On the other side, a number of specialized custom simulators exist such as TOSSIM [16], Avrora [20] and Atemu. Those simulators put focus on specific aspects of WSN. They provide models for e.g. node level performance, energy consumption, operating system level behavior. Some simulators focus on lower level behavior, while some are focused on high level behavior. Since models of one aspect do not cover the interaction and interference between aspects, simulation results are not satisfying for system-level analysis and DSE. The cross level simulator COOJA [18], provides a combination of low-level and high-level behavior prediction in a single simulation environment. It provides better insight in system-level behavior; however, it is too slow to be applied efficiently in DSE. Recently, a WSN simulation framework for the Omnet++ simulator was developed – MiXiM [13]. That framework includes many libraries and detailed models for lower layers. MiXiM presents good solutions for simulating WSN behavior at various system levels. The speed of MiXiM is acceptable for analyzing system behavior and obtaining insight in the interaction between system parts, but still it is not sufficiently fast for DSE.

There is a clear tradeoff between accuracy of the performance estimation and the scalability of the simulator. In order to increase scalability, most simulators provide a limited amount of detail and focus modeling efforts on specific system parts (MAC model, radio model, application model). In this paper, a layered system model is proposed. Each layer is modeled with a simple model with a limited level of detail. Reducing the level of detail raises a risk of oversimplification. In [14], some simplistic axioms were proven wrong (circular transmission area, perfect reception …). Here we have sought to avoid those mistaken simplifications.

3. WSN CASE STUDY

A temperature mapping application has been implemented on the MyriaNed WSN platform, developed at DevLab in collaboration with VU, both in the Netherlands [21]. In order to avoid collisions and to coordinate actions over the shared channel, an energy efficient TDMA based MAC protocol is used. No directed routing exists on the top of this MAC layer. Rather, data dissemination is realized according to a probabilistic replication scheme. This section gives a brief description of the temperature mapping system, and then defines the design space.

3.1 Temperature Mapping

A temperature mapping application is a representative environment monitoring application, with a goal to collect several sensor readings, from a set of sensors deployed at known positions. Generally, environment monitoring applications assume a large number of sensors (hundreds) spread over some area (e.g. a potato field) in order to collect data over a long period (months to years). Furthermore, it can be assumed that network topology is more or less static and that data need to be collected at regular intervals. For a slow changing environment phenomenon (such as temperature), low data rates are acceptable. Data dissemination is based on a gossiping metaphor.

gMac is a MAC protocol specifically designed to service gossiping-based traffic developed in collaboration between DevLab and Delft University [1]. gMac is a TDMA-based protocol with a broadcast message delivery that provides collision-free communication. Each node follows a communication schedule that is computed synchronously for all nodes in the network. Typically, the radio communication is the most energy consuming part of the MAC protocol. In order to save energy, gMac creates communication schedules such that idle communication is minimized. Ideally, a node listens only in those slots when it expects packets from its neighbors. Furthermore, gMac limits the number of transmissions per frame (TDMA period) to only one packet. gMac scheduling provides for each node to choose its own transmission slot so that in a two-hop neighborhood no node transmits in the same slot. Thus, gMac avoids the hidden terminal problem. This solution is similar to the LMAC protocol [11]. In gMac, the number of packets which a node can receive per frame is limited (the number of receive slots per frame is a gMac parameter). Schedules are calculated such that just a part of the neighbor nodes are able to receive a transmission. The challenge in configuring a gMac protocol is in the trade-off between energy consumption and bandwidth provisioning.

SharedState is a scheme for probabilistic storage and replication of the common-interest data in a WSN [8]. It provides a "fair" distribution of data items to all nodes in the network. It is a state-of-the-art gossip-based application and thus suitable to be implemented on top of gMac. This combination provides high robustness (gossip), while keeping energy consumption low (gMac). In this paper, we consider a slightly modified SharedState protocol, adapted to serve as a temperature mapping application. Each node has a local cache where data items from all nodes in the network are stored. The data item has a part that contains information about the temperature value, and a part that comprises information about the place and time of the measurement (ID and timestamp). Each time a new packet arrives, SharedState updates the node-local cache. Only one reading per place can be stored in the cache, so when a node receives an item with the same ID and a newer timestamp, the old item is replaced. Once per frame, SharedState randomly chooses items to be transmitted. Since gMac provides only one transmission slot per frame and only 5 data items fit in one packet, SharedState fills a packet with its own local measurement and 4 random items from the cache.

A gossip protocol spreads information in a manner similar to the spread of a virus. It results in a probabilistic system in which information "diffuses" from its source into the network. Time necessary for an "infected" node to spread its data item to its neighbor is not fixed, but a random variable and depends on the system properties. Thus, "infection" spreading paths are random (Figure 1). We are looking for system models that capture this mechanism, and provide means to analyze system properties.

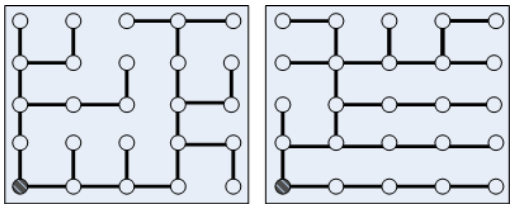

Figure 1. Information diffusion in gossip networks (item starts from the lower left node)

3.2 System Metrics and Parameters

Before starting with modeling, the problem at hand is defined in more detail. System-level quality metrics and parameters that provide an insight in the system behavior have to be selected.

Typical WSN metrics are latency, reliability, and lifetime. **Latency** denotes the average time necessary for a data item to propagate to all nodes in the network (for data items that do reach all nodes). **Reliability** denotes the average ratio of data items delivered successfully. Due to the probabilistic distribution scheme, a newer version of the same data item may be distributed faster than an old one. In that case, since a node keeps just the latest version in its cache, the old one is considered to be a lost data item (not successfully delivered). **Lifetime** denotes the time it takes until nodes in the network deplete their energy source and thus depends on the energy consumption. The gMac protocol provides predictable energy consumption. The number of active radio slots per frame is fixed, leading to fixed energy consumption per frame.

The chosen configuration parameters define the design space, and thus their number should be kept reasonably small in order to keep the design space explorable. Four configuration parameters which provide different effects on the system metrics are chosen here: i) sensor sampling period (T_{SMP}), ii) transmission power (P_{TX}), iii) frame duration (T_{FRAME}), and iv) the number of reception slots per frame (N_{RX}). The last two parameters have to be homogenously set over the network (network level), while the others allow heterogeneous settings over the network (node level).

4. WSN PERFORMANCE EVALUATION

In this paper, we describe a fast and accurate performance evaluation technique for WSN. The key assumption is that the WSN allows accurate probabilistic abstractions. A layered system model is proposed, and probabilistic abstractions are used for describing the behavior of layers and interactions between them. Abstractions should be simple, but have to capture essential elements that have an effect on the system behavior. For demonstration purposes, we choose a case study that allows simple probabilistic models.

Our performance evaluation approach is divided into two steps: i) creating an accurate system model and ii) calculating performance based on that model. In the first step, the system dimensions (parameters and metrics) are defined. The properties of the layers and their interactions are analyzed, through low level simulations. Atomic models describing behavior of basic elements of a system are constructed. For the case study, three atomic models are considered: the radio model, the MAC model and the SharedState model. Those atomic models are connected in a single system model (Figure 2a). In the second step, based on this system model, two different simulators are developed. For minimizing simulation time, statistical techniques are used. Metrics are estimated based on the simulations (Figure 2b). The two steps are analyzed in the next two sections. The focus is on latency and reliability. Lifetime is not considered. Due to the predictable energy consumption of MyriaNED, lifetime can be computed without the techniques proposed in this paper. If energy consumption varies due to the stochastic nature of a WSN, we expect our method to capture lifetime as well, although this needs experimental evaluation.

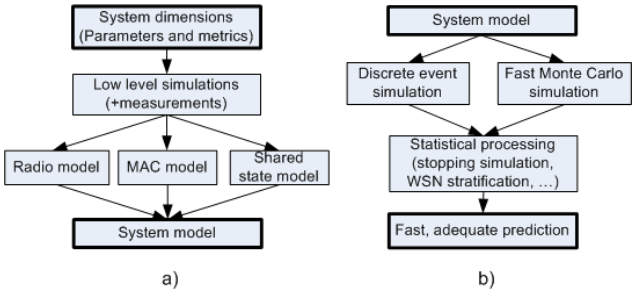

Figure 2. Performance evaluation. a) System model b) Simulation

5. Layered System Model

Design-space exploration requires fast, yet accurate system-level models. In this section, a layered model is proposed as an adequate solution to this problem.

The general concept is to decouple the effect that a system's parts have on the overall performance, to create simple models for each of those parts, and then to analyze the interaction between those parts. Based on that analysis, the atomic models are combined into a system model. Traditionally, a system can be divided into layers, where each layer assumes a collection of conceptually similar functions that provide services to the layer above it and receives service from the layer below it. Here, accordingly, the properties of the SharedState, gMac, and radio propagation layers are analyzed and modeled. Although it might be possible to capture properties of each system layer in an analytical model, it is highly problematic in general to combine those atomic models in one analytical model which captures the essential system properties. Reasons for this lie in the system size and non-homogenous settings over the network. In such a system, the number of the various interactions between nodes and variation in those interactions over the network has a stronger effect on the system (un)predictability than the complexity of the individual system parts.

As a case study for the proposed layered model, the temperature mapping application as described in Section 3 is used. In the rest of this section, the layered model is described in detail, atomic parts are defined and modeled, and system behavior is analyzed.

5.1 Shared state model

Each node in the network measures a local temperature. Furthermore, each node creates and maintains its own temperature map of the whole WSN. The number of data items a node keeps in its local cache is equal to the number of sensor nodes (*Ns*). In each time frame, SharedState selects 5 items that should be transmitted

in the next packet. One of the selected items is always the item with local measurements, while the other 4 items are randomly selected from the cache (see Figure 3).

The probability that a local item X is selected from the cache is 1. In the case that X is not a local item, there are (Ns-1) items in the cache that can be selected with equal probability. SharedState selects 4 of those items. So, the probability that item X is selected is as follows:

$$P_{SS\_SEL}(X) = \begin{cases} \dfrac{4}{N_S - 1} & X \neq local\ item \\ 1 & X = local\ item \end{cases} \quad (1)$$

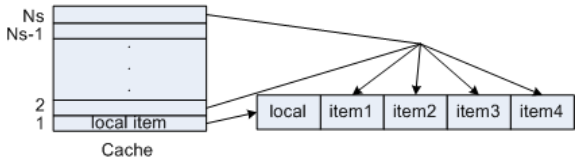

Figure 3. SharedState selection process

5.2 The gMac Model

TDMA-based protocols are relatively simple for modeling. gMac scheduling provides for an absence of collisions and 2-hop interference. Communication between two nodes at the gMac layer is modeled by the probability of a successful reception.

In order to save energy, gMac calculates schedules where in each time frame just a part of a node's neighbors listen for a transmission. In each time frame, a different subset of neighbors listens in. When a node transmits a message, each subset of neighbors has equal probability to have their receivers turned on. Thus, the probability that a node listening to receive items is equal to the probability that its subset is active in that frame:

$$P_{gMac} = 1/N_{SCH} \quad (2)$$

where N_{SCH} denotes total number of subsets.

5.3 Radio Link Model

Two nodes are assumed to be connected if they are able to exchange messages in more than a sporadic fashion. Many factors may affect connectivity between nodes. For purpose of this work, we limit the problem dimension to two important factors: the location of the nodes and their transmission power levels.

Fluctuations in the quality of the radio channel have a big impact on the system behavior. As our focus was not on radio modeling, the radio model used is rather simple. However, this radio model is sufficiently flexible such that it can be easily calibrated with measurements from a test bed. First, based on a path loss model, the maximal radio range is calculated and a connectivity map is obtained. The link quality between each pair of the nodes in the radio range is modeled by a packet reception rate – PRR. The nature of observed PRR in experimental setups is discussed in [22] and the model used in this work is created according to those guidelines. With e.g. measurement data from a test bed, our model could be calibrated to best match test bed properties.

To avoid possible problems with the gMac protocol, we assume (at current) that communication links are symmetrical. Also, the implemented probabilistic radio model does not yet take into account temporal correlation between fluctuations in link quality and the "on-off" nature of the radio channel. However, when measurement data becomes available, we plan to improve the radio link model accordingly.

5.4 Interactions – System Description

A WSN system may consist of tens or hundreds of interacting nodes. Depending on the local conditions, each node may experience different behavior. The ultimate goal is to understand the effect that local changes have on system-level performance, to find models that capture the most important interactions and interferences, and to combine that insight with the atomic models to create an integrated WSN system model.

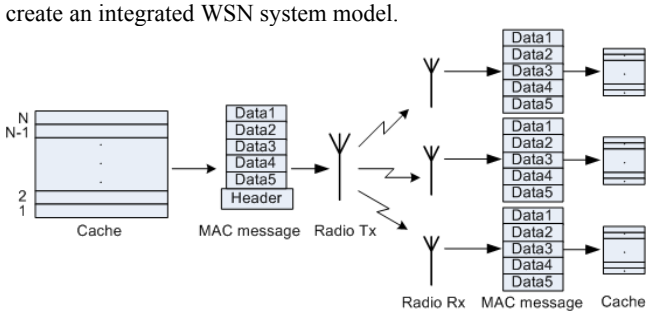

Figure 4. Description of the system behavior

For our case study, nodes measure in regular intervals a local temperature and pass on that information to the dissemination protocol. SharedState creates a local temperature item, includes it in the packet together with 4 other items randomly chosen from the cache and passes the packet to the gMac layer. Then, in the appropriate time slot, gMac turns on the transmitter and broadcasts the packet. All nodes within radio range have a chance to receive the packet. However, due to location dependent variations in the quality of the radio channel, a packet may not be delivered successfully to all neighbors. Furthermore, according to the communication schedule, some of the nodes might have their receiver turned off at that moment. When a node does receive a packet, gMac extracts the data items and hands them to SharedState. SharedState then updates its local cache. A schematic description of the system behavior is given in figure 4. The next section describes system level models, as part of the description of our performance analysis techniques.

6. FAST SIMULATION METHODS

Two common approaches for the performance evaluation of a WSN are analysis and simulation. An analytical approach usually provides the fastest execution time. However, it is often not possible to find accurate analytical models. On the other hand, functional system simulation is often too slow and thus not feasible for DSE. In this paper, we propose simulation techniques that work fast and yet evaluate system performance with adequate accuracy. In the remainder of this section, first a low-level MiXiM simulator is described; then a more abstract discrete event simulator based on a probabilistic graph model is introduced; and finally we propose a fast, Monte Carlo simulation approach, with the highest level of abstraction.

6.1 MiXiM Simulations

MiXiM is a simulation framework developed for wireless and mobile simulations in OMNET++. OMNET++ is a "public-source, component-based, modular and open-architecture simulation environment" [13]. The combination of OMNET++ and MiXiM creates a powerful discrete event simulator suitable for all kinds of low-level simulations. The MiXiM framework consists of numerous libraries containing channel, connectivity, mobility, obstacle, and communication protocol models. Simulation time of

the MiXiM framework is acceptable for protocol debugging and system analysis, but it is far too large for DSE. Results obtained by the MiXiM simulator, can be used as a reference for estimating accuracy of other, more abstract simulators.

We implemented the complete temperature mapping system in the MiXiM framework. Specific parts of the temperature mapping system, such as gMac and SharedState are modeled to fit in the MiXiM environment. The simulator is adjusted to record data of interest and to estimate the metrics. This MiXiM simulator was employed to calibrate the atomic models, and as a reference to evaluate our proposed fast simulation methods (see Section 7).

6.2 PGM Based Simulation

The performance of a WSN can be estimated based on a Probabilistic Graph Model (PGM). A PGM captures the elementary relationship between atomic models and integrates them into a system model.

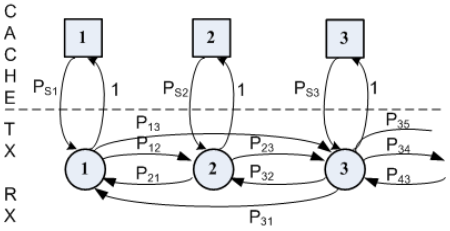

Figure 5. Probabilistic graph model (PGM)

In a PGM, a wireless sensor network is represented by a weighted graph (Figure 5). Each sensor node is represented with two vertices: the bottom one corresponds to the transceiver; the top one represents the cache. Links connecting vertices represent vertex-vertex relations. A cache vertex is connected only to an adjoining transceiver. The probability that a data item is chosen for transmission is P_{SS_SEL}, and thus the weight of the edge going from a cache to a transceiver is P_{SS_SEL}. For each two transceivers, a connecting edge exists only if they can communicate. The weight of that edge corresponds to the probability that a packet sent by transceiver Y is successfully received by transceiver Z (P_{YZ}). A packet is successfully received if it is propagated successfully through the radio channel (probability P_{radYZ}) and if gMac listens in that time slot (probability P_{gMac}). Thus, the edge weight by per-transmission independent probabilities is given by:

$$P_{YZ} = P_{gMac} \cdot P_{rad\,YZ} \,. \qquad (3)$$

In the PGM, simulation is performed in turns, where a turn corresponds to a gMac frame. Each node is represented by a virtual cache, which is simply the representation of the actual cache in the WSN node. In each turn, node-specific operations are simulated and the virtual cache is updated. The stochastic simulator "flips a coin" according to the given probability in order to select items from the cache and check transmission success. If the transmission is successful, the simulator compares the version of the received item with the version of the item in the cache. If it is older, the received item is discarded. The simulator logs each change in the cache content. From the logged data, the average latency and reliability are calculated as follows. For each pair (item, destination node) the average delivery time is computed. The latency is calculated as the average value of all pair delivery times. Similarly, for each item the average delivery ratio (the number of the nodes that successfully receive the item) is computed. The reliability is calculated as the average value of all item delivery ratios.

6.3 SVGM Based Simulation

The PGM requires every node to be evaluated every turn. To increase simulation speed, a new approach is proposed, based on Monte Carlo simulation. Here, the system performance is evaluated based on a shortest-path abstraction of a stochastic weighted graph. Observe that the probability of selecting a specific item from the cache (1), just as the probability of successful transmission of that item (3) does not depend on the propagation of other items in the network. So, instead of simulating a complete system at once, the distribution of each item is simulated separately. Then, based on the principle of superposition, system performance is evaluated based on individual results. This section first introduces the stochastic model that forms the basis of our method, and then continues with the shortest-path abstraction used to compute the metrics.

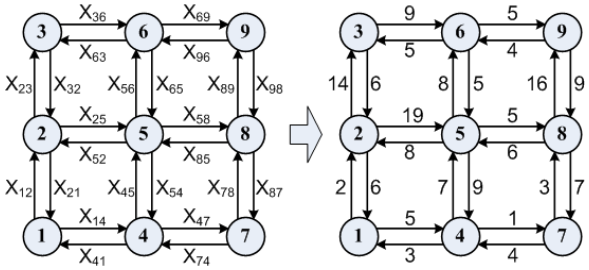

Figure 6. Stochastic-variable graph model (SVGM)

6.3.1 The Stochastic-Variable Graph Model

A WSN is described with a stochastic-variable graph model - SVGM. Vertices represent nodes, edges connectivity between two nodes, while edge weights are stochastic variables describing propagation time (delay until an item is selected and successfully transmitted) between two nodes. Repeated random sampling is used to estimate the stochastic behavior of the system. In each simulation round, one instance of the SVGM is sampled. In this context, a simulation round thus corresponds to the distribution of one item version through the network, until it is either delivered to all nodes, or overwritten everywhere with a newer version. Weights are sampled randomly, according to adjoined stochastic variables (see Figure 6).

When a node selects an item from the cache, the item is broadcasted and each of the neighbors has a chance to receive it. Thus, there is a certain spatial correlation between times necessary to deliver an item to a node's neighbors. This delivery time is determined by the number of attempts needed before a packet is transmitted successfully (considering the packet loss probability) and the time between those attempts (depending on the random process of selecting items from the cache). Spatial correlation between delivery times for a node with three neighbors is illustrated in Figure 7. In the example, an item is (re)transmitted in slots 3, 7, 10, and 15. Neighbor1 successfully receives the first transmission; neighbor2 receives the third, while neighbor3 receives the item only after four transmissions. Intervals between retransmission are the same for all neighbors; thus, this is the spatial correlation between delivery times. The time between retransmissions is distributed according to the geometric distribution F1(x), with success probability p1=P_{SS_SEL}. Similarly, the number of attempts before an item is transmitted successfully from node Y to node Z is distributed according to the geometric distribution F2(x) with success probability p2=P_{YZ}.

The stochastic variable X_{YZ} in Figure 6, denotes the time that the item spends in the cache of node Y before it is delivered to node Z. Based on the previous analysis, a general strategy for random sampling of the stochastic variable is defined. First, for each neighbor node, the number of necessary retransmissions $N_R(Z)$ is sampled from geometric distribution F2(x). Then, intervals between retransmissions are sampled from F1(x). The spatial correlation between intervals is taken into account. Thus, all neighbors have the same retransmission intervals. Delivery time to node Z is calculated by summing up the first $N_R(Z)$ retransmission intervals. Sampling a random number according to a geometric distribution is done by the procedure described in [9].

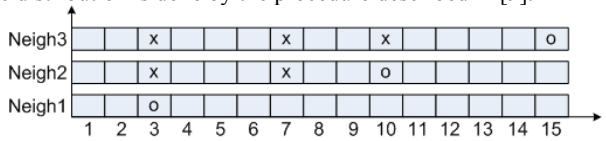

Figure 7. Spatial correlation between delays

SVGM depends on the models behind the stochastic variables. In our case study, stochastic variables denote delay. Considering, as an alternative example, a contention based MAC, stochastic variables can for example be the number of retransmissions. In that case, SVGM can be used for calculating latency, but also for energy consumption.

6.3.2 Shortest-Path Abstraction

The time necessary to deliver an item from its origin X to some node Y is the sum of the times that an item needs to propagate over each segment of the path from X to Y. The time necessary to propagate over one segment (between two nodes) is described with the stochastic variables of the SVGM. In order to emulate the random behavior, in each simulation round, a different weighted graph is sampled from the SVGM. The path that an item traverses from X to Y is the shortest (in this case the fastest) path connecting those two nodes in the weighted graph instance. The shortest paths are calculated using Dijkstra's algorithm [4]. The delay is calculated as the shortest path length in that round. Then, the system latency in that round is calculated as an averaged sum of the lengths of the shortest paths among all pairs. Figure 8 gives shortest paths from node 2 to all other nodes, based on the graph of Figure 6.

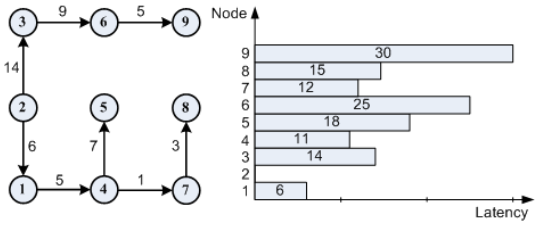

Figure 8. Calculating latency from node 2 to all other nodes

The average system latency is obtained after N_{SIM} simulation rounds, with N_{SIM} the simulation length. The calculated shortest path lengths are then random variables driven from the underlying stochastic variables for link propagation delays. Thus, if the number of simulation rounds is large enough, the result of the shortest-path abstraction will be statistically the same as the result of PGM discrete-event simulation (the time estimate will converge, in stochastic terminology, *almost surely* to the same value).

The shortest-part abstraction described so far does not include information about lost packets and reliability. In order to handle packet loss, Dijkstra's algorithm has to be modified slightly. The chance of packet loss depends strongly on the sampling period, T_{SMP}. The more often items are sampled, the higher is the chance that some items will be overtaken by newer versions and therefore are considered lost. An item version is lost at a certain destination node if any of the later versions propagates faster and arrives earlier at that destination node. In order to evaluate packet loss, knowledge about the future (propagation of later item versions) is required. Thus, the simulation is performed in reversed chronological order, i.e., the last simulation round is executed first, then the round before, and so on. The packet loss condition can now be made more precise.

Let $T_{XZ}(i)$ denote the latency of item X, version i, where Z is the destination node. Then, item X, version i, is not lost if and only if in the round corresponding to $T_{XZ}(i)$ item version $i+k$ (with $k>0$) is not in the cache of node Z and not in the cache of the node that transmits the item to Z, say node U. This can be written as:

$$T_{XZ}(i) < T_{XZ}^{*}(i+1) \;\&\; T_{XZ}(i) < T_{XU}^{*}(i+1) \qquad (4)$$

with $T_{XY}^{*}(i+1)$ the earliest delivery time of any later data from node X to node Y, computed from the latency for these item versions increased with the difference in the sampling times:

$$T_{XY}^{*}(i+1) = \min_{k \in N} \left(kT_{SMP} + T_{XY}(i+k) \right) \qquad (5)$$

The loss condition is illustrated in Figure 9 (with T_{SMP}=10, X=0). For the given situation, condition $T_{02}(i) < T_{01}^{*}(i+1)$ is not satisfied. Thus, item version i will never arrive at node 2 via node 1. Since in this example there are no other paths to node 2 than via node 1, item version i is lost for node 2, and consequently for node 3.

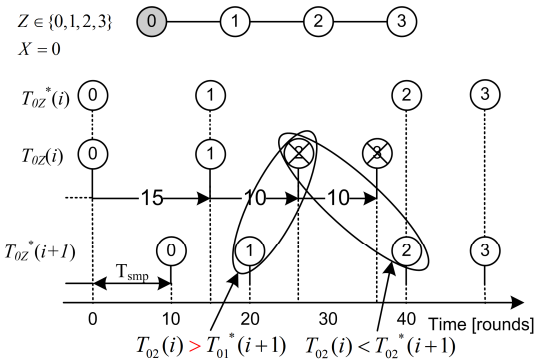

Figure 9. Evaluating loss conditions for reliability calculations

Our modified Dijkstra's algorithm (see Figure 10) takes three inputs: the weighted graph G sampled from the SVGM, the source node X, and the latency list $T_{XZ}^{*}(i)$. As result, Dijkstra's algorithm provides information about latencies (shortest paths), reliability (the number of times loss condition (4) is satisfied) and the updated latency list (as an input for the next simulation round). All relevant "future" data necessary to evaluate the loss condition is comprised in the latency list, $T_{XZ}^{*}(i)$, $Z \in \{1, \ldots, N_s\}$ from the previous round.

In the original Dijkstra's algorithm, in each round, the node with the smallest delay so far is chosen, and its neighbors are updated with new delays. In the modified version, neighbors are not updated automatically. The algorithm first checks the loss condition (4). If the item is not lost, the node's neighbors are updated. Not updating the neighbors has the same effect as preventing that version of the item to propagate via that neighbor. The modified Dijkstra's algorithm updates the latency list at the end of

each simulation round; delays corresponding to the lost items are replaced with delays of items that overtook them. The updated list (Figure 9) is used as the input for the next simulation round.

```
Dijkstra(G, X)
1. Initialize-Single-Source(G, X)
2. S ← 0
3. Q ← V[G]
4. while Q ≠ 0
5.     do u ← Extract-Min(Q)
6.        S ← S ∪ {u}
7.        for each vertex v ∈ Adj[u]
8.            do Relax(u, v, G, )
```

```
Dijkstra(G, X, T_XZ*(i), Z ∈ {1,..., N_S})
1. Initialize-Single-Source(G, X)
2. S ← 0
3. Q ← V[G]
4. while Q ≠ 0
5.     do U ← Extract-Min(Q)
6.        S ← S ∪ {U}
7.        for each vertex V ∈ Adj[U]
8.            if (T_XV(i−1) < T_XV*(i) & T_XV(i−1) < T_XV*(i))
9.                do Relax(U, V, G, )
10.           else item does not arive at node v through node u
11. update T_XZ*(i−1), Z ∈ {1,..., N_S}
```

Figure 10. Dijkstra's algorithm and its modification

7. EXPERIMENTAL EVALUATION

We performed a set of experiments with the goal to estimate the accuracy and speed of the proposed method, as well as its scalability. Since SVGM simulation results converge to the PGM discrete-event simulation results, and the former simulations are faster than the latter, we focus on the comparison between low-level MiXiM simulations and SVGM simulations.

The relative method accuracy is estimated by comparing latency and reliability results obtained from SVGM simulations with reference results from MiXiM. Simulations are performed for different topologies and configurations. The total configuration space is too large to be simulated exhaustively. For our evaluation, we performed simulations for homogenous distributions of system parameters over the network. Figure 11, presents results for four KxK grid topologies, $K \in \{5,6,7,8\}$, with $T_{SMP} = \lfloor K^2/2 \rfloor$. The comparison is performed only for smaller network dimensions to overcome the very long running times of MiXiM simulations. In order to observe the behavior of our method in the conditions when reliability calculations can be verified, the temperature sampling periods are chosen to provide significant numbers of lost packets. Each experiment consists of a certain number of subruns, N_{SR}, where each subrun includes the propagation of N_{SI} versions of the item, repeated for each item-generating node. The final result is the average value from all subruns, where we want to obtain at least a 95% confidence level. For SVGM simulations, we conservatively set N_{SI}=1000. The statistical significance of the result based on N_{SR} subruns was calculated based on the estimation of long-run sample averages technique [23]. For all SVGM experiments, 30 subruns were run, which turned out to be enough to get statistically strong estimates in all cases. On the other hand, the speed limitation of MiXiM prevents large sets of simulations within reasonable time. Thus, the MiXiM results are based on 20 subruns covering 250 versions of each item. MiXiM results nevertheless have similar statistical strength (95% confidence level). Relative prediction errors of the SVGM simulations are given in Figure 11. SVGM results are close to MiXiM results, showing at most 1.1% difference.

Table 1. Running times for MiXiM and SVGM simulators

Sim/Grid	5x5	6x6	7x7	8x8	9x9	10x10
MiXiM [min]	26	52	94	153	256	402
SVGM [s]	0.77	1.42	2.54	4.85	6.85	9.85

For the same set of the experiments, running times of the two simulators are compared in Table 1, where SVGM simulation times are shown in seconds, and MiXiM simulation times in minutes. Results are given per subrun, to allow a direct comparison between the two simulation methods. As expected, the SVGM simulator is much faster. The SVGM simulation speeds are adequate for DSE, showing simulation speeds in the order of seconds.

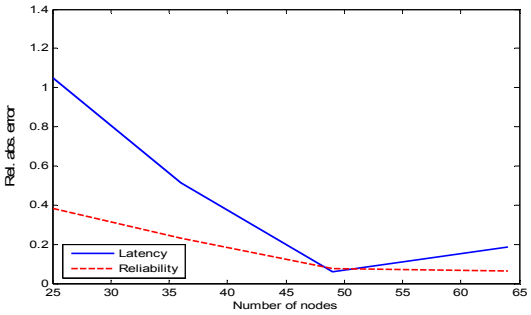

Figure 11. Accuracy of prediction

An interesting aspect is to examine SVGM computational complexity. Each simulation round is dominated by the runs of the modified Dijkstra's algorithm. The algorithmic complexity of Dijkstra's algorithm implemented with binary heaps is: $O(E \lg N_S)$, where E is the number of edges in the SVGM model, and N_S the number of WSN nodes. This procedure is repeated for each item-generating node and each data item version that this node generates. The total computation complexity is therefore $\underline{O}(N_{SR} \cdot N_{SI} \cdot N_S \cdot E \cdot \lg N_S)$. Large values for N_{SR} and N_{SI} increase statistical strength, but on the other hand increase experiment duration as well. A more detailed investigation of good values for N_{SR} and N_{SI} is a part of our future research plans. The first experiments with larger networks indicate that $N_{SR} \cdot N_{SI}$ does not increase with network size. The practical computation complexity, is therefore expected to be $\underline{O}(N_S \cdot E \cdot \lg N_S)$, indicating that the method scales well to larger networks.

In summary, the SVGM method is demonstrated on a case study with gossip-based communication. It has been shown that the method can handle probabilistic distribution mechanisms well. The short running time and adequate accuracy show that this method is well suited for use in DSE.

8. CONCLUSION

In this paper, an integral WSN modeling approach is presented, aiming to improve the understanding and prediction capabilities of WSN system level properties. The proposed highly abstract layered system model has several important advantages over the current state-of-the-art models. It provides a fast and scalable way to estimate WSN performance from a system perspective. Speed of the performance evaluation is of high importance for DSE (design-space exploration). The proposed abstractions allow that DSE is performed in a reasonable time frame. The scalability of the proposed method is adequate; it can be efficiently used for networks with several hundreds of nodes. Its scalability can be further increased using additional statistical techniques. The modular approach makes the models easily tunable for changes in system parts, including other applications.

The layered system model and the PGM simulation technique are generally applicable for different scenarios of WSN systems. PGM allows highly abstracted discrete-event simulation and is

thus applicable to any system described with probabilistic models. Its speed and accuracy depend on the complexity and accuracy of the underlying abstractions. SVGM provides even higher levels of abstraction, modeling a random process through the stochastic variables. SVGM is a basis for fast Monte Carlo simulation. Generally the SVGM provides that the propagation of individual data items can be simulated (not modeled) independently of the propagations of other items. For the gossiping scenario, the SVGM is coupled with the shortest-path abstraction with stochastic variables as edge weights in the graph. This method has been analyzed in our case study, but can be adapted for other probabilistic application as well.

In a real deployment, it is expected to have many dynamic changes and stochastic effects, leading to unpredictable propagation patterns. Some of those effects are described in the case study: random selection of data to be sent, random choice of active neighbors, variations in radio link quality throughout the network. Some others will be considered in future work: time-dependent link quality variations, interference, moving obstacles, node mobility. SVGM applicability depends on the possibility to describe those random effects concisely and accurately by stochastic variables (stochastic edge weights). If it is possible to do so, SVGM based techniques can show their full potential and significantly reduce simulation time.

An additional reduction in the running time may further improve the scalability of the method. Some initial experiments indicate that system properties can be estimated accurately by analyzing behavior of just a few nodes instead of all nodes. The idea is to exploit regularity in the network topology and apply stratification techniques. We plan to investigate this in future work. In parallel, we plan to develop DSE techniques such as those of [17] based on our models. Finally, such fast system-level property evaluation may enable model-driven adaptation techniques that aim to maintain quality of service in dynamic circumstances.

9. REFERENCES

[1] Anemaet, P.A. 2008. Distributed G-MAC: A flexible MAC protocol for servicing gossip algorithms. Master's thesis, Technical University of Delft, The Netherlands

[2] Antoniou, P. and Pitsillides, A. 2007. Wireless Sensor Networks Control: Drawing Inspiration from Complex Systems. Med-Hoc-Net 2007, Ionian Academy, Corfu.

[3] Boukerche, A., Nakamura, A., Loureiro, A. 2008. Algorithms for Wireless Sensor Networks: Present and Future. Chapter in Algorithms and protocols for wireless sensor networks, John Wiley & Sons.

[4] Cormen,T., Leiserson, C., Rivest, R., Stein, C. 2001. Introduction to Algorithms (2nd ed.). MIT Press.

[5] Chen, S. and Yang, N. 2006. Congestion avoidance based on lightweight buffer management in sensor networks. IEEE Transaction on Parallel and Distributed Systems, vol.17, no.9, pp.934–946.

[6] Ferentinos, K.P. and Tsiligiridis, T.A. 2007. Adaptive design optimization of wireless sensor networks using genetic algorithms. In Journal of Computer networks, volume 51, pages 1031-1051. Elsevier.

[7] Ganesan, D., Estrin, D., Woo, A., Culler, D., Krishnmachari, B., Wicker, S. 2002. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical Report CS TR 02-0013, UCLA.

[8] Gavidia, D., van Steen, M. 2008. A probabilistic replication and storage scheme for large wireless networks of small devices. In Proceedings 5th IEEE Int'l Conf. Mobile and Ad Hoc Sensor Systems (MASS).

[9] Gentle, E. 2003. Random number generation and Monte Carlo methods. Springer.

[10] Hoes, R., Basten, T., Tham, C.K., Geilen, M., Corporaal, H. 2007. Analysing QoS trade-offs in wireless sensor networks. In 10-th ACM Int. Conf. on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM). ACM.

[11] Hoesel, L. van and Havinga, P. 2004. A lightweight medium access protocol (LMAC) for wireless sensor networks, in 1st Int. Workshop on Networked Sensing Systems, Tokyo, Japan.

[12] Kuorilehto, M., Kohvakka, M., Suhonen, J., Hämäläinen, P., Hännikäinen, M., Hämäläinen, T.D. 2007. Ultra-Low Energy Wireless Sensor Networks in Practice, John Wiley & Sons.

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

[14] Kotz, D., Newport, C. and Elliot, C. 2003. The mistaken axioms of wireless-network research. Dartmouth College Computer Science Technical Report TR2003-467.

[15] Langendoen, K. 2008. Medium Access Control in Wireless Sensor Networks. Chapter in "Medium Access Control in Wireless Networks", Nova Science Publishers.

[16] Levis, P., Lee, N., Welsh, M. 2003. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. First ACM Conference on Embedded Networked Sensor Systems (SenSys 2003), Los Angeles, California, USA.

[17] Nabi, M., Blagojevic, M., Basten, T., Geilen, M., Hendriks, T. 2009. Configuring Multi-Objective Evolutionary Algorithms for Design-Space Exploration of Wireless Sensor Networks. PM2HW2N'09, Tenerife, Spain. ACM.

[18] Osterlind, F., Dunkels, A., Eriksson, J., Finne, N. and Voigt. T. 2006. Cross-level sensor network simulation with cooja. In Proceedings 2006 31st IEEE Conference on Local Computer Networks, pages 641—648. IEEE.

[19] Stankovic, J. 2007. Wireless Sensor Networks. Chapter in Handbook of Real-Time and Embedded Systems, CRC Press.

[20] Titzer, B., Lee, D.K. and Palsberg, J. 2005. Avrora: Scalable Sensor Network Simulation with Precise Timing. Proceedings of IPSN'05, Fourth International Conference on Information Processing in Sensor Networks, Los Angeles.

[21] Wateren, F. van der. 2008. The art of developing WSN applications with MyriaNed. Tech. report, Chess Company.

[22] Zuniga, M., Krishnamachari, B. 2004. Analyzing the transitional region in low power wireless links. In Proc. of IEEE Sensor and Ad Hoc Communications and Networks (SECON), Vol. 1, pp. 517-526, Santa Clara, CA. IEEE.

[23] Zwillinger, D., Kokoska, S. 1999. Standard probability and statistics tables and formulae. Chapman&Hall.