Search-based Stress Testing of Wireless Network Protocol Stacks

Matthias Woehrle
Embedded Software Group
Delft University of Technology
Delft, The Netherlands
m.woehrle@tudelft.nl

***Abstract*—The operation of wireless network protocol stacks is heavily dependent on the actual deployment of the system and especially on the corresponding network topology, e. g., due to channel contention. The nature of wireless communication does not allow for a-priori determination of network topology; network-defining metrics such as neighbor density and routing span may drastically differ for various deployments. Therefore, it is a difficult problem to foresee and consider the large number of possible topologies that a system may run on during protocol stack development. We propose to use an automated approach for searching topologies for which a protocol stack exhibits particularly poor quantitative performance. We formulate stress testing of protocol stacks on specific topologies as a multi-objective optimization problem and use an evolutionary algorithm for finding a set of small topologies that particularly stress the protocol stack of a wireless network. For searching the topology space, we present novel problem-specific variation operators and show their improvements on search performance in case studies. We showcase our results on stress testing using two protocol stacks for wireless sensor networks.**

***Keywords*-Testing, Software Testing, Wireless Networks, Wireless Sensor Networks**

I. Introduction

Wireless networks typically have strict requirements on quantitative properties of the protocol stack: examples include the power consumption of a Medium Access Control (MAC) protocol and the packet yield of a data collection protocol. However, the operation of network and MAC protocols strongly depends on deployment characteristics, especially on the specific topology of the deployment. This is because the topology affects which nodes can directly communicate with each other; moreover, since the wireless medium is shared among neighboring nodes, communication may interfere locally and transmissions may collide, e. g., due to hidden terminals. Determining the effect of specific topologies on the protocol stack is non-trivial and can typically only be determined by experimentation, mainly because simplistic assumption on radio models can lead to misleading or incorrect results [22]. Hence, for a protocol developer it is interesting to see which specific topologies negatively affect the performance measured by some quantitative properties.

To this end, *this work proposes an automated stress testing method by evaluating quantitative properties of wireless network protocol stacks considering the diverse set of topologies a system may run on*. Our work presents a search-based testing method that allows us to find network topologies that exacerbate the operation of a protocol stack. In particular, our approach focusses on testing Quality-of-Service (QoS) metrics for wireless (sensor) network protocol stacks. We strive to find network topologies that stress a network's operation and minimize its QoS. To achieve this, we formulate an optimization problem for finding topologies that result in low QoS. Our search-based testing approach requires multiple (at least two) objectives: *(i)* a measure of the size of the network and *(ii)* one or more QoS properties. Since the relation of QoS properties to deployment topology is non-trivial and typically unknown, we rely on stochastic black-box optimization techniques.

The main objectives of this paper can be summarized as follows: *First*, to identify why a protocol stack performs poorly on a given topology, we need to analyze the corresponding executions. The analysis and debugging is obviously easier if the network is smaller since we only need to look at a smaller number of interactions between nodes. Therefore, we minimize topologies searching for the smallest topology exhibiting the same characteristics. *Second*, QoS properties are vital for evaluating the system running on a given topology. Properties such as packet yield, latency (end-to-end delay) and energy consumption are used to determine QoS for data collection protocols [16], code propagation protocols [26] and MAC protocols [4] amongst others. The second objective is to find the smallest topology for a given (lower) QoS.

The main contributions of this paper are the following:

- We propose a search-based approach for stress testing wireless network protocol stacks and show how to formulate testing w. r. t. different network topologies and the quantitative properties of the protocol stack as an optimization problem.
- We investigate the use of multi-objective evolutionary algorithms (MOEAs) to search the state space and propose novel variation operators for the problem domain.
- We show two case studies using network protocols from the wireless sensor network domain: we test the

packet yield of a multi-hop data collection protocol, the Collection Tree Protocol (CTP) [16], and the QoS of a code propagation protocol, called Trickle [26].

The paper is structured as follows. We start by presenting background and related work in Sec. II. We detail on the modeling of the optimization problem for search-based stress testing in Sec. III, followed by a description of the proposed search-based testing strategy based on evolutionary algorithms in Sec. IV. Sec. V presents the experiments on a MOEA-based approach by comparing it to a "naive" random search-based approach. Sec. VI discusses what we can learn from topology-based testing w. r. t. the efficiency and scalability of a protocol stack, before concluding the paper with a summary in Sec. VII.

II. Background and related work

In the following, we focus on wireless sensor networks as one class of wirelessly networked systems. A wireless sensor network is a distributed system composed of resource-constrained embedded devices called sensor nodes. A sensor node typically consists of a radio, a microprocessor, some storage, one or more sensors, and a limited energy source. Although sensor nodes are quite limited by themselves they can collaboratively monitor a spatial phenomenon, such as permafrost in the Alps [5], with a high spatial and temporal resolution. Deployments of sensor networks have strict requirements on quantitative performance and are often deployed in harsh environments. Most wireless sensor network protocols, e. g., MAC protocols such as XMAC [9] and EM-MAC [34], are evaluated based on quantitative, non-functional properties such as energy-efficiency. QoS is thus vital for testing as well as for the design process, e. g., in design space exploration [20]. We investigate the effect of network topology on the QoS of two well-known protocols for wireless sensor networks. The first protocol is a multi-hop data collection protocol, the Collection Tree Protocol (CTP) [16]. Data collection is one of the mainstream application scenarios in wireless sensor networks. Data sensed at individual sensor nodes is routed to the gateway node that provides access to the data to the external world, e. g., by connecting to the Internet. For data collection, QoS is mainly determined by the packet yield, i. e., how much data from the sensors is received at a common gateway node. The second protocol used as a case study is Trickle [26], a code propagation algorithm based on "polite gossiping". The Trickle protocol is used to disseminate software update information when the embedded software running on every sensor node needs to change. A gateway node indicates a new version of the software; in turn, all sensor nodes need to be notified of the software update, i. e., the update needs to be disseminated from the gateway node to the whole network.

Verification of sensor networks operation is sought-after; however, exhaustive methods are severely hampered for wireless sensor networks by state space explosion. Firstly, executions of a distributed system (on a fixed topology) feature a large number of next-state transitions. Secondly, the set of of possible topologies results in a large number of initial states. Both the state space and the number of initial states are exponential in the number of sensor nodes. Therefore, up to now, testing and verification of sensor networks has mostly focussed on efficient verification of functional properties, e. g., using domain-specific model checking [30], symbolic executions [33] and randomized depth-first search state space exploration [27]. Exceptions are the work of Ballarini et al. [4] on QoS of Medium Access Control and our previous work on testing the power consumption of sensor nodes [36]. However, all of these works rely on a fixed topology. In this work, we take a fundamentally different approach by investigating the impact of topologies. This allows us to:

1) find the smallest topology for each QoS performance level and
2) explore the efficiency and scalability of protocol stacks w. r. t. their QoS.

Note that the first goal (of test input minimization) is similar to work on test case minimization such as delta debugging by Zeller et al. [39] and unit test case minimization [24], despite the fact that we do not have qualitative, but quantitative properties. Hence, we formulate stress testing as a search-based test problem [18], [29]. Similar to the work by Li et al. [28] we prioritize test cases in that we focus on topologies that exacerbate the operation of the protocol stack. We present an evolutionary algorithm-based approach for test input generation. This is not a new idea – Xanthakis et al. [37] were the first to propose evolutionary testing in the 1990s. Multi-objective problem formulations have been studied previously, e. g., for branch coverage by Harman et al. [17], mutation coverage [23] and test case generation from extended finite state machines [38]. Ferrer et al. [13] compare different approaches for the multi-objective test data generation, thereby providing a good overview of state-of-the-art (multi-objective) evolutionary algorithms. In contrast the contribution of this paper is to use a multi-objective problem formulation for wireless networks to minimize QoS and topology size; for this problem domain, we present and evaluate new crossover operators.

Clark [11] previously proposed search-based testing using simulated annealing and genetic algorithms for protocol security. We focus on quantitative, nonfunctional properties of protocols and how these are (negatively) affected by network topology. Afzal et al. [1] present an overview of search-based testing of non-functional properties. Out of the works surveyed in [1], two use search-based testing of QoS of web services using genetic algorithms: Canfaro et al. [10] test QoS-aware service composition, while Di Penta et al. [12]

test service level agreements. In contrast to these works, we look at wireless networks that are dependent on their deployment, i. e., environmental conditions. This work is loosely related to Nguyen et al. [31] in that we want to investigate the environmental effects on an autonomous distributed system. However, we specifically focus on stress testing of distributed systems, similar to the work by Garousi [15], yet focus particularly on the impact of the topology on the protocol stack of wirelessly networked systems.

The approach presented in this paper is unique in that it relates non-functional properties to network topology size (and structure). This allows us to find the smallest topologies with a given QoS in order to (*i*) determine the efficiency of protocol stacks as well as (*ii*) to find small topologies that exhibit a particular (poor) behavior. For the first goal, we realize that systems may suffer from scalability issues. As a network grows and contains certain topological features such as high network density and a large depth, the QoS of a network protocol may deteriorate. We describe this further in the following section.

Note that search-based stress testing is inspired by our previous work on optimizing node deployments for wireless sensor networks [35]. In contrast to optimizing node deployments, where we aimed at maximizing the network's connectivity, the search-based topology testing formulation takes the opposite approach and tries to minimize the utility (w. r. t. QoS) of a given network.

III. Approach

Topology testing relies on formulating a search problem. We start by describing the model of a network topology and proceed with detailing on the optimization problem.

A. Topology model

We model a deployed network as a graph G=(V,E) where the vertices represent wireless sensor nodes and edges represent communication links. Since the space of all possible graphs is very large and most graphs do not have a physical realization, i. e., there is no network deployment that results in such topologies, we use *random geometric graphs* [32] in two-dimensional space to approximate the set of feasible deployments. In a nutshell, a random geometric graph is a random undirected graph drawn on a bounded region; we use $[0, 1) \times [0, 1)$. Nodes are placed at random uniformly and independently on this bounded region. Two nodes u, v can communicate iff the geometric distance $d(u, v)$ between them is at most a communication threshold t, i. e., $d(u, v) \leq t$. In this paper we simplify the model for simulation and abstract from link qualities and directionality as this facilitates the comparison of different approaches as described in Sec. V.

Using the topology model we evaluate a given protocol stack and determine its different QoS metrics. Our formulation is generic and allows us to use different methods of evaluation given the topology model, e. g., simulations and an analytical model. In this work we focus on simulations as detailed in the case studies.

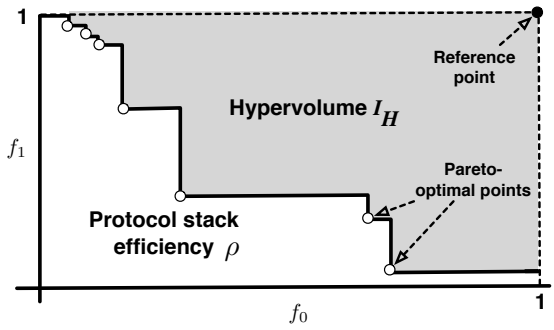


Figure 1. Pareto front based on the Pareto-optimal points (circles) for two objectives f_0 and f_1, a single QoS property f_1 and the topology size f_0. We can either use the protocol stack efficiency or the hypervolume for formulating the optimization problem.

B. Stress testing using topologies

We are interested in stress testing and therefore in the worst-case performance of a protocol stack given different network topologies. Hence, we are looking for network topologies that result in worst performance w. r. t. QoS. Intuitively, for a given topology size s we have to find the topology with minimal QoS within the set of all possible topologies. Obviously, the set of all possible topologies is exponential in s. In the following, we detail on the corresponding (multi-objective) minimization problem that we use to search for worst-case topologies.

In particular, we want to quantify the *protocol stack efficiency* ρ w. r. t. a set of performance metrics $F, |F| = m$ and a measure of the size of the network topology. Formally, we want to *minimize* $\vec{y} = \vec{f}(G) = [f_1(G), \ldots, f_m(G)]^T$, where G denotes the topology (graph) as described above and $f_i : (V, E) \to \mathbb{R}$ denotes a function for QoS metric i achievable by the protocol stack given a topology G. Let us denote the size of a network as function $f_0(G)$. Without loss of generality, let us assume that we normalize the size of a network such that $f_0 \in [0, 1]$. Similarly we normalize each QoS property $f_i \in F$ such that $f_i \in [0, 1]$. We use Pareto-optimality as classically defined w. r. t. dominance [40]. For our minimization problem, a point G_1 dominates G_2 iff:

$$\forall i \in \{0, \ldots, n\} : f_i(G_1) \leq f_i(G_2) \wedge$$
$$\exists j \in \{0, \ldots, n\} : f_j(G_1) < f_j(G_2)$$

Basically, a non-dominated point is Pareto-optimal. We refer to the set of Pareto-optimal points as the Pareto front. The Pareto front of QoS performance points allows us to define a metric of efficiency of a protocol stack.

As shown in Fig. 1, we can formulate the dual problem to protocol stack efficiency using the hypervolume indicator I_H [14]. In particular we calculate the hypervolume w. r. t. to a reference point [40]. In this work we choose the point of

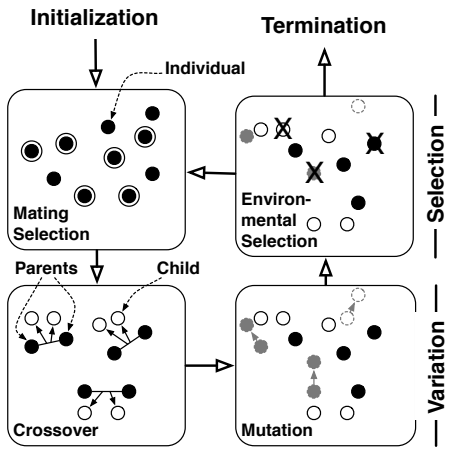


Figure 2. General view of multi-objective evolutionary algorithms [7].

optimal efficiency $\vec{1}$, where $|\vec{1}| = m+1$. Intuitively as shown in Fig. 1, the hypervolume is determined by calculating the volume between the front of Pareto-optimal points and the reference point. It follows that $\rho = 1 - I_H$. We use the hypervolume also to evaluate the performance of the variation operators; note that we use the terms hypervolume, hypervolume indicator and I_H synonymously.

We want to minimize the efficiency of the protocol stack, i. e., we are interested in the worst-case operation by searching for topologies that particularly stress the protocol's operation. Hence, we are looking for maximizing the hypervolume in the evaluations. In addition to protocol stack efficiency as described by ρ, we can also investigate *protocol stack scalability*: We may inquire whether QoS of a protocol stack drops with increasing the number of nodes as shown in Fig. 1. We investigate scalability on a parametrized protocol stack in Sec. VI.

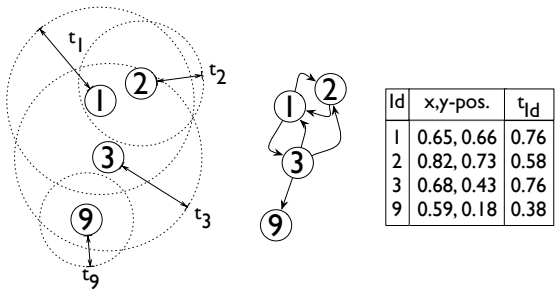
IV. A MOEA-BASED APPROACH

We evaluate the use of multi-objective evolutionary algorithms (MOEAs) for searching the topology space. We detail on the MOEA-based approach in the following and compare it to a "naive" random search-based strategy. We present a general view of MOEAs [7] in Fig. 2 to outline the nomenclature used in the following.

A. MOEA for topology testing

We use an off-the-shelf MOEA, the Simple Indicator Based Evolutionary Algorithm (SIBEA) [8] as it is provided in the PISA framework [6]. We adapt initialization, and the variation operators, i. e., crossover and mutation, to the new search space, where one crossover and the mutation operators are similar to our previous work in [35].

1) Representation: An individual represents an entire wireless sensor network as a set of nodes and their communication capabilities. More precisely, an individual stores a set of nodes; for each node it records an identifier, the cartesian coordinates and a communication threshold. Based on the



Id	x,y-pos.	t_{Id}
1	0.65, 0.66	0.76
2	0.82, 0.73	0.58
3	0.68, 0.43	0.76
9	0.59, 0.18	0.38

Figure 3. Variable length representation of an individual: A topology consists of a set of nodes. Each node description consists of an ID, its cartesian coordinates and a communication threshold.

communication threshold, the simulation model determines the communication links between nodes whether based on signal strength [35] or using a unit-disk graph model as described in Sec. III-A. A simple example in Fig. 3 shows that the topology model is general and may result in asymmetric links. Since the number of nodes is one of the optimization criteria, we explicitly allow variable-length representations, i. e., sensor networks with a varying number of nodes. Please note that we use the physical placement merely to generate valid topologies – in essence, we are only interested in the graph structure of a topology.

2) Constraints: Some generated networks do not represent practical deployments. We use constraints to remove topologies that should not be considered. In the case studies, we use two constraints on the generated networks: (*i*) Topologies must be connected, i. e., there should be no partitions. (*ii*) We additionally use an upper threshold on node density set to 40. Note that more problem-specific constraints may be easily added.

3) Initialization: In the initialization step we generate topologies from the family of random geometric graphs. Hence, we have two parameters for a graph: (*i*) the number of nodes n and (*ii*) the communication threshold t, i. e., t is initially the same for all nodes. We select t from a uniform distribution between $[0, 1)$, yet reject any resulting topology that does not satisfy the constraints described above. We use a common communication threshold to initialize all links of the network, yet after variation these can be heterogeneous for a network given that our variation operators allow us to add new nodes with a different t and the crossover combining two topologies containing nodes with (potentially) different t's. In order to allow for a better comparison, the generation of random geometric graphs is the same as the initialization for the MOEA approach, i. e., we randomly generate random geometric graphs as detailed above.

4) Crossover: In the following, we investigate several crossover operators. We start with the cut operator [35].

Cut: The crossover operator cuts the physical region of two parents and recombines them. Different to its use in [35] we do not need to consider the convex hull of a space of

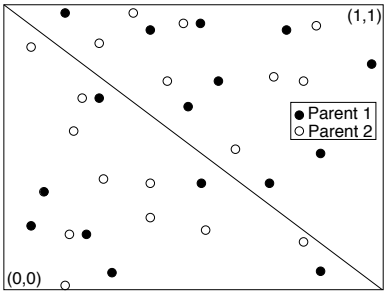


Figure 4. Cut crossover operator from [35] in a unit square.

interest, but perform the crossover directly on the unit square a random geometric graph is placed into as shown in Fig. 4.

We ran micro-benchmarks to see the performance of the cut-crossover in isolation. While the cut operator previously worked well when all topologies try to find a (more or less) even distribution, here we often see that it fails to produce good offsprings. As we can see from the results of the EA, i. e., the resulting hypervolume shown in Figure 6(a) on the left, the cut operator performs quite poorly for our search problem. Katagiri et al. [21] also observed in their work that small changes on graphs may be better for the fitness of the crossover. Thus, we additionally investigate two variants of extracting a connected subgraph of the parents and replacing it with a corresponding connected subgraph in the other parent.

Fixed neighborhood swap (*swap*): The *swap* operator selects a node from one parent topology and determines the corresponding closest node in the other parent topology. In the example depicted in Fig. 5, the selected nodes are annotated in light gray. These nodes, as well as all neighboring nodes of these selections (dark gray in the figure), constitute the *swap set*. In turn these node sets are swapped between the respective topologies. The selected (light gray) nodes are swapped so that their position is adjusted to match the position of the corresponding node in the other parent. All other nodes in the swap set (dark gray) are moved with the same displacement as the selected nodes. As an example, consider nodes 1,2 and 3 in the figure and their respective positions $x_i, y_i, i \in \{1, 2, 3\}$. The new position of node 3 x'_3 when placed into parent 2 is calculated as:

$$x'_3 = \begin{cases} x_3 - x_1 + x_2 & \textit{if } 0 \leq x_3 - x_1 + x_2 \leq 1 \\ 0 & \textit{if } x_3 - x_1 + x_2 < 0 \\ 1 & \textit{if } x_3 - x_1 + x_2 > 1 \end{cases}$$

y'_3 follows correspondingly.

Flexible neighborhood swap (*flexible*): *Flexible* is a variation of the swap crossover. It maintains the node selection functionality and merely changes the swapping behavior described above. Instead of moving the selected nodes with a fixed displacement, it tries to maintain the topology in the neighborhood by formulating the neighborhood relation of

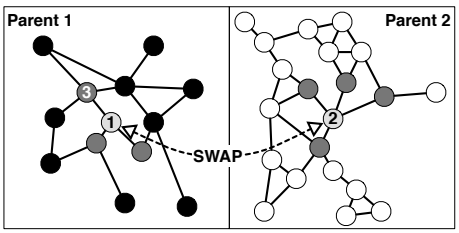


Figure 5. Swapping the neighborhoods of two corresponding nodes (light gray) and their direct neighbors (dark gray) from each parent topology.

the selected (light gray) nodes as constraints. As an example, it tries to place node 1 in the parent 2 topology in a way that it maintains exactly 3 neighbors (and not more); to accomplish this, the transmission radii of the nodes might need to be adapted as well. The flexible crossover uses a constraint solver in order to find a physical (coordinate) embedding for the swapped nodes respecting the neighborhood constraints.

5) Mutation: We evaluate two different mutation operators on topologies described in our previous work [35] with one difference: The operators mutate multiple nodes c in a topology at once. This is due to the fact that mutation on a single node does not result in high hypervolume. We performed several experiments to determine how many nodes should be mutated. In this work, we always mutate at least one node plus a (rounded) random number sampled from an exponential distribution. As a result, the number of nodes to change c is determined as $c = 1 + \textit{round}(x)$, with the random sample x drawn from $f_{exp}(x, \lambda)$ with $\lambda = \frac{1}{2}$.

Topology change: The topology change mutation operator adds or removes c random nodes from the topology. The probability of adding and removing is $\frac{1}{2}$.

Position change: The position change adds gaussian variation to the position (both x and y position) of c nodes with $\mathcal{N}(0, \sigma)$. The standard deviation is set to $\sigma = \frac{r_i}{2}$, where r_i is the transmission radius of the selected node i.

V. Case studies

In the following we present two case studies focussing on the network protocols CTP and Trickle. We detail on the performance of the evolutionary algorithm and the proposed crossover operators by comparing it to a random search strategy.

A. Experimental setup

For the evolutionary algorithm, we use 40 individuals per generation as in our previous work [35]. The maximum number of generations is set to 100. This is obviously a trade-off since evaluation of individuals is expensive and the hypervolume improvements by running additional generations diminish as we will show in the results in Fig. 8(b). We compare the results of the evolutionary algorithm-based approach to a random search generating 4000 (random) topologies. We compare the search strategies based on the

hypervolume from 10 individual runs with different initial seeds.

The whole testing framework is implemented in Python and relies on NetworkX (http://networkx.lanl.gov/) for all graph operations on topologies. The embedding operation used for the flexible crossover operator is formulated as a constraint satisfaction problem and solved using ECLiPSe [2]. For each topology representation we generate the topology's adjacency matrix and model (existing) links as perfect. Additionally, the simulation model always generates symmetric links. We use TOSSIM [25] for simulation, a discrete-event simulator that is part of the TinyOS operating system distribution [19]. TOSSIM allows us to simulate actual sensor node applications based on a simple radio propagation model. We build a test application for each specific protocol stack. On the network layer we use the standard implementation of the protocols in TinyOS. Note that the MAC layer is fixed in the experiments, since TOSSIM only provides a fixed CSMA-based MAC. While our focus is on the network layer we must consider that the selection of a CSMA MAC protocol obviously influences the behavior of the complete protocol stack. Note that we need to consider that our approach necessitates a large number of evaluations, i. e., simulations. This means that simulations must be short, since we need to perform multiple runs (we repeat each simulation of a topology four times) for testing the behavior under different conditions. We evaluate runs based on the hypervolume as described in Sec. III-B.[1]

B. Protocol stacks

In the following we describe the two protocol stacks under test. We explain the application that we use in order to test the stack as well as the specific optimization problem formulations.

1) CTP: CTP [16] is a tree-based, multi-hop data collection protocol for wireless sensor networks. Data is collected from a set of nodes to one (or more) gateway nodes. Nodes generate routes to the gateway node using a routing gradient based on hop distance and link quality. In this paper we test how many packets that each sensor node sends actually arrive at the gateway node. To that end, we use a data collection application, where each node sends 50 packets. Packets are spaced $5s$ apart; intermediately the node samples its sensors. The gateway node keeps track on how many packets it received from individual nodes.

We use the number of nodes n as the first optimization criterion:

$$f_0 = \frac{n}{100}$$

We normalize the number of nodes, such that $f_0 \in [0, 1]$. Note that we could easily look at additional metrics such as the number of communication links, average in- or out-degree of nodes, etc.

As a second minimization criterion we use the packet yield averaged over all nodes $j \in \{1, \ldots, n\}$ and averaged over the number of experiments $i \in \{1, \ldots, 4\}$:

$$f_1 = \frac{1}{4} \sum_{i=1}^{4} (\frac{1}{n-1} \sum_{j=1}^{n} \frac{r_j^i}{50})$$

where r_j is the number of packets that the gateway node (node 0) received from node j. The superscript i indicates a simulation run.

2) Trickle: The second protocol stack under test is based on Trickle [26], a probabilistic (code) dissemination protocol. One node, i. e., the gateway node, is informed of an update of the embedded software and advertises it to the network. Nodes that hear an update propagate it further. We are interested in the control flow of updates, i. e., whether all nodes are informed that an update is waiting to be applied and how fast the nodes are notified of the update. An important mechanism of Trickle is that a node forwards a code update to other nodes if it has received an update itself, yet the number of updates it received is not larger than a preset redundancy constant r (we set r to 3). This is used to control excessive messaging leading to high contention in the network. However, in certain topology configurations this constant may preclude certain nodes from receiving updates - for a more detailed discussion of this problem we refer the interested reader to Mottola et al. [30].

For testing Trickle, we used its standard implementation in TinyOS used in a dissemination application. After an initialization period of $50s$, the application generates in total 10 updates on the gateway node. Updates are spaced $10s$ apart. We again use the normalized number of nodes as the first optimization criterion, i. e., $f_0 = \frac{n}{100}$. As the second optimization criterion we use the following dissemination criterion mapped to an interval $[0, 1]$:

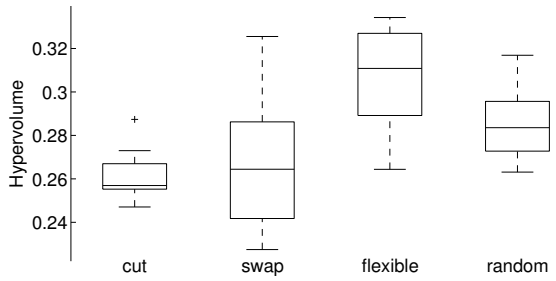
$$f_1 = e^{-\frac{1}{100} * d_{avg}}, \textit{ where } d_{avg} = \frac{1}{4} * \sum_{i=1}^{4} (\frac{\sum_{j=1}^{n} d_j^i}{n-1}))$$

and d_j^i is the dissemination delay of a code update to node j for packet i. This criterion evaluates dissemination latency but also scores lost updates. As described above some nodes might not receive certain updates at all. While we could add a third optimization criterion for minimizing lost updates, we include these directly into the dissemination criterion by setting a maximal dissemination delay of $d_{max} = 500$ (sec) for "lost" updates.

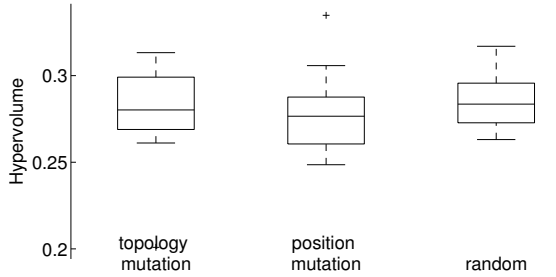
C. Micro-benchmarks on CTP

We first perform micro-benchmarks and determine the performance of individual crossover and mutation operators individually. Note that we evaluate the results of the operators based on the resulting hypervolume of the search and always compare to the "naive" random search strategy.

[1] As a reference point, for the implementation we actually choose $(1.01, 1.01)$ resulting in a maximal hypervolume indicator value of ≈ 1.0201.



(a) Comparison of the crossover operators.



(b) Comparison of the mutation operators.

Figure 6. Hypervolume distribution for variation operators. Boxplots of 10 runs each.

Crossover operators: Figure 6(a) summarizes the performance of the different crossover operators. As mentioned before the cut operator does not perform well for our optimization problem with a median hypervolume of 0.257. The swap operator is only slightly better than the cut operators but exhibits a high variance. The flexible crossover performs considerably better than all other approaches achieving a median hypervolume of 0.311. Therefore, we selected the flexible crossover operators for the following experiments.

Mutation operators: Figure 6(b) shows that both mutation operators perform similar to a random search. Hence, we maintain both mutation operator to see their performance in combination with the flexible crossover operator.

Random search: All in all, the random generation of topologies works fairly well. Note that this is also beneficial for the MOEA formulation as the random generation uses the same approach as the initialization of the MOEA to allow for a fairer comparison.

D. MOEA evaluation

The final set of experiments investigates the performance of combining the mutation operators with the flexible crossover operator, where mutation and crossover have both an equal probability of 50%. Fig. 7 shows that the evolutionary algorithms perform considerably better than the random approach. Indeed the median hypervolume for the random front is 0.284, while the hypervolume of the flexible crossover combined with position mutation achieves a mean hypervolume of 0.339. Note that the crossover combined with topology mutation perform slightly worse, yet shows

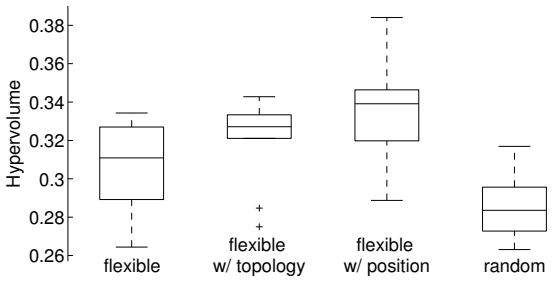


Figure 7. Combined boxplots of selected operators for the MOEA-based approach for 10 runs each.

less variance in the experiments. All in all, we can see that adding mutation (moderately) improves the results.

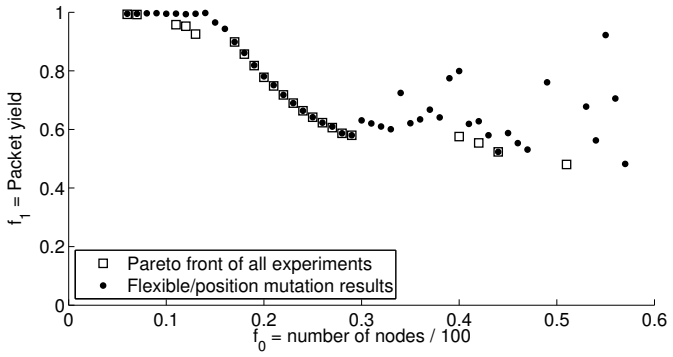
While the MOEA-based approach is clearly better than random search, the question remains on how good it really is. This is dependent on the maximum possible hypervolume, which is determined by the actual protocol stack efficiency ρ of the data collection protocol stack and in particular CTP. We aggregate all Pareto-optimal results from all different runs that we performed (all MOEA runs as well as the random runs) to approximate this maximum. We show the resulting front (squares) in Fig. 8(a) that gives a hint at the "real" efficiency ρ of the data collection application in our configuration. This combined front actually achieves a hypervolume of 0.404, indicating that we can further improve our search strategy in the future. As a comparison the figures also shows the results of all runs of the flexible crossover combined with position mutation as circles.

Fig. 8(b) shows the evolution of the hypervolume indicator over generations for all runs of the flexible crossover combined with position mutation. We see that the hypervolume slowly converges. We stop at 100 generations. Obviously this is a trade-off: Since results improve only marginally, we trade-off further gains with the computational overhead of further simulations.

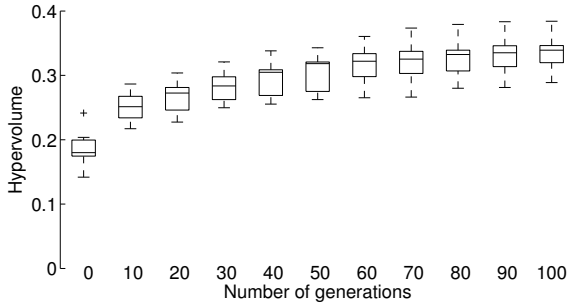
E. Validation using Trickle

Up to now, we have validated our approach only using CTP. Thus, the question is whether the results hold for very different kinds of protocol stacks and properties. While we cannot answer this question in general we performed some experiments using the Trickle protocol. Note that we expect to find small topologies with poor protocol stack performance as the small redundancy constant precludes the protocol to spread the updates reliably as described above. This means that the dissemination optimization criterion is mainly influenced by lost updates.

Fig. 9 shows that the MOEA-based approach finds such small bad topologies in each individual search – the median hypervolume found is 0.948, the minimum is 0.933. In contrast, the random search has a median value of merely 0.859 and a maximum of 0.921. As a result, we can see



(a) Pareto front (squares) for all runs combined, i.e. squares are Pareto-optimal points for all evaluations performed. The circles are the results found by all runs of the evolutionary strategy using the flexible crossover with a gaussian position mutation.



(b) Hypervolume of the MOEA generations (in steps of 10). Boxplots of 10 runs.

Figure 8. Runs of the evolutionary algorithm with the flexible operator and position mutation.

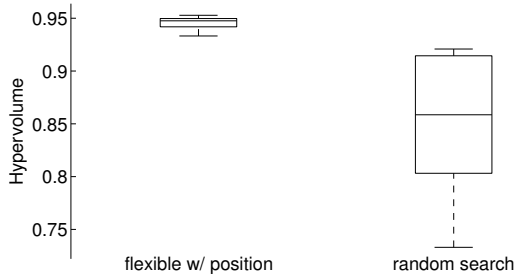


Figure 9. Search-based testing results for the Trickle protocol for 10 runs each.

that for the Trickle testcase the MOEA-based approach is considerably better and more robust than the random search.

F. Threats to validity

Our search-based formulation for stress testing protocols is very generic and can be applied to any protocol stack that provides quantitative properties for evaluation. The results on the performance of the specific mutation and crossover operators may however be dependent on the specifics of the protocol stack under test. We addressed this concern by considering two different protocol stacks, relying on different communication mechanisms (unicast versus broadcast messaging) and considering different quantitative properties. Nevertheless, these are just two samples from the vast design space of network protocols, so further experiments are needed to investigate the performance of the variation operators on other protocol stacks. Moreover, since stochastic search algorithms are affected by chance, we repeated each experiment 10 times with different random seeds in order to fairly compare the different approaches. For tuning the evolutionary algorithm, we relied on our experience and previous experiments on optimizing node deployments [35] and performed micro-benchmarks to explore the performance of the different variation parameters. However a full parameter tuning, e. g., as described in [3], is out of scope of this work. As an example, we only investigated using only mutation, only crossover and combination with equal 50% probabilities. Obviously further combinations may be explored for potentially better results.

VI. Protocol stack scalability

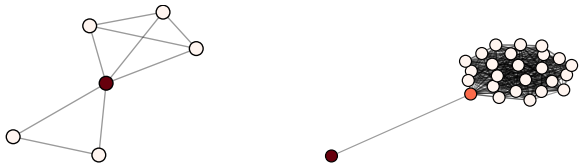
As a final step of our approach we want to show an intuitive application of protocol stack scalability (cf. Sec.III-B). We first discuss results on the data collection application followed by a detailed investigation for the Trickle protocol stack. All experiments are performed using the MOEA-based approach with the flexible operator combined with position mutation as described above.

A. CTP

For CTP we clearly see in Fig. 8(a) that the packet yield decreases with size. Figures 10(a) and 10(b) illustrate the difference; Figures 10(a) is a topology with a good yield, while the yield for the topology in Fig. 10(b) is quite poor. The major problem is that the neighbor density considerably increases in the larger topology. Most nodes in the larger topology are located closely together (in the upper right corner of Fig. 10(b)) resulting in high congestion and thus low packet yield. Note that we already mitigated contention to some extent by putting a constraint on neighbor density. There are several further possibilities to remedy this situation: (*i*) Either we must specify that the data collection application is targeted for low-density topologies and thus further decrease the network density constraint or (*ii*) improve the scalability of the protocol stack by mitigating congestion – a prime candidate would be the MAC layer. Note that in our experiments the MAC protocol is fixed to CSMA by using the TOSSIM simulator.

B. Trickle

We investigate the effect of different parameterizations of the Trickle protocol by comparing their respective performance. In particular we investigate the influence of the re-



(a) Topology with 6 nodes and packet yield 0.995.

(b) Larger topology with 26 nodes and packet yield 0.623

Figure 10. Topologies for CTP exploration on the Pareto front. Darker node color indicates proximity to the gateway node, i. e., the darkest node is the gateway. Note that nodes are arranged to show connectivity, rather than the physical location determined by the MOEA.

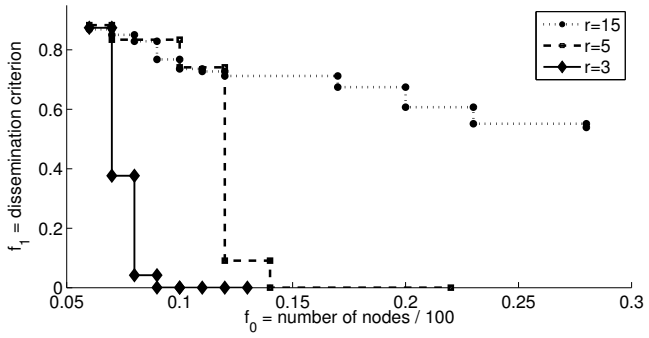


Figure 11. Protocol efficiency for Trickle for different parameterizations of redundancy constant r.

dundancy constant r. We test Trickle for $r = \{3, 5, 8, 10, 15\}$ and identify the differences in scalability due to increasing the redundancy constant. Figure 11 visualizes the scalability of Trickle for three of these parameterizations. As we can see, the efficiency of the protocol stack, and likewise the scalability, drastically improves when increasing r. Below we show the results for the protocol efficiency ρ dependent on different values of r:

r	3	5	8	10	15
ρ	0.053	0.091	0.432	0.526	0.583

The improved efficiency is not surprising - more messages are being sent, thereby trading of reliability with energy expenditure. Increasing r improves scalability: As Fig. 11 shows that for $r = 5$ the dissemination criterion drops to 0.091 for 12 nodes, while for $r = 15$ the protocol maintains stable operation with a dissemination criterion of at least 0.539. However, the benefit of increasing r decreases. The results for $r = 10$ and $r = 15$ are similar, but the energy expenditure using $r = 15$ is considerably larger than for $r = 10$. Therefore, in the future we need to additionally investigate the energy-efficiency of the stack as a third objective. We leave this evaluation as future work.

VII. Summary

This paper presented an approach to perform stress testing of wireless network protocol stacks considering the influence of the deployment settings w. r. t. the topology. We demonstrated how to formulate an optimization problem that considers the size of the topology as well as quantitative properties of the protocol stack. We proposed a search-based approach using multi-objective evolutionary algorithms to find small topologies that exhibit poor QoS. For our specific problem domain of network topologies, we proposed new crossover operators and benchmarked their performance. We evaluated our approach using two different and prominent wireless sensor network protocol stacks. Our testing approach allows us to compare protocol stacks w. r. t. their efficiency and scalability; we showed this for different parameterizations of the Trickle protocol. Additionally, we showed with two case studies that minimizing topologies with poor Quality-of-Service facilitates the understanding of protocol performance under diverse operating conditions.

Acknowledgements: Matthias Woehrle is supported by the Dutch Technology Foundation STW and the Technology Programme of the Ministry of Economic Affairs, Agriculture and Innovation. The author thanks Koen Langendoen, Andreas Loukas, Philipp Glatz and the anonymous reviewers for their valuable feedback.

VIII. References

[1] W. Afzal, R. Torkar, and R. Feldt. A systematic review of search-based testing for non-functional system properties. *Inf. Softw. Technol.*, 51:957–976, 2009.

[2] K. R. Apt and M. Wallace. *Constraint Logic Programming using Eclipse*. Cambridge University Press, 2007.

[3] A. Arcuri and G. Fraser. On parameter tuning in search based software engineering. In *Proc. 3rd Int'l Conference on Search based software engineering*, pages 33–47, 2011.

[4] P. Ballarini and A. Miller. Model checking medium access control for sensor networks. In *Proc. 2nd Int'l Symposium on Leveraging Applications of Formal Methods, Verification and Validation (ISoLA 2006)*, pages 255–262, 2006.

[5] J. Beutel, S. Gruber, A. Hasler, R. Lim, A. Meier, C. Plessl, I. Talzi, L. Thiele, C. Tschudin, M. Woehrle, and M. Yuecel. PermaDAQ: A scientific instrument for precision sensing and data recovery in environmental extremes. In *Proc. 8th ACM/IEEE Int'l Conference on Information Processing in Sensor Networks (IPSN 2009)*, pages 265–276, 2009.

[6] S. Bleuler, M. Laumanns, L. Thiele, and E. Zitzler. PISA—A Platform and Programming Language Independent Interface for Search Algorithms. In *Proc. 2nd Int'l Conference on Evolutionary multi-criterion optimization (EMO 2003)*, volume 2632 of *LNCS*, pages 494–508, 2003.

[7] D. Brockhoff. *Many-Objective Optimization and Hypervolume Based Search*. 2009. PhD thesis at ETH Zurich.

[8] D. Brockhoff and E. Zitzler. Improving SIBEA rvolume-based Multiobjective Evolutionary Algorithms by Using Objective Reduction Methods. In *Proc. congress on Evolutionary Computation (CEC 2007)*, pages 2086–2093, 2007.

[9] M. Buettner, G. V. Yee, E. Anderson, and R. Han. X-mac: a short preamble MAC protocol for duty-cycled wireless sensor networks. In *Proc. 4th Int'l Conference on Embedded networked sensor systems*, SenSys '06, pages 307–320, 2006.

[10] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An approach for QoS-aware service composition based on genetic algorithms. In *Proc. 7th Conference on Genetic and*

evolutionary computation (GECCO 2005), pages 1069–1075, 2005.

[11] J. A. Clark and J. L. Jacob. Protocols are programs too: the meta-heuristic search for security protocols. *Information and Software Technology*, 43:891–904, 2001.

[12] M. Di Penta, G. Canfora, G. Esposito, V. Mazza, and M. Bruno. Search-based testing of service level agreements. In *Proc. 9th Conference on Genetic and evolutionary computation (GECCO 2007)*, pages 1090–1097, 2007.

[13] J. Ferrer, F. Chicano, and E. Alba. Evolutionary algorithms for the multi-objective test data generation problem. *Software: Practice and Experience*, 2011.

[14] M. Fleischer. The measure of pareto optima applications to multi-objective metaheuristics. In *Proc. 2nd Int'l Conference on Evolutionary multi-criterion optimization (EMO 2003)*, pages 519–533, 2003.

[15] V. Garousi. A genetic algorithm-based stress test requirements generator tool and its empirical evaluation. *IEEE Trans. Softw. Eng.*, 36:778–797, 2010.

[16] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection tree protocol. In *Proc. 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, 2009.

[17] M. Harman, K. Lakhotia, and P. McMinn. A multi-objective approach to search-based test data generation. In *Proc. 9th Conference on Genetic and evolutionary computation (GECCO 2007)*, pages 1098–1105, 2007.

[18] M. Harman and P. McMinn. A theoretical and empirical study of search-based testing: Local, global, and hybrid search. *IEEE Transactions on Software Engineering*, 36:226–247, 2010.

[19] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. In *Proc. 9th Int'l Conference on Architectural support for programming languages and operating systems (ASPLOS-IX)*, pages 93–104, 2000.

[20] R. Hoes, T. Basten, C.-K. Tham, M. Geilen, and H. Corporaal. Quality-of-service trade-off analysis for wireless sensor networks. *Perform. Eval.*, 66:191–208, 2009.

[21] H. Katagiri, K. Hirasawa, J. Hu, and J. Murata. Comparing some graph crossover in genetic network programming. In *Proc. 41st SICE Annual Conference*, volume 2, pages 1263 – 1268, 2002.

[22] D. Kotz, C. Newport, R. S. Gray, J. Liu, Y. Yuan, and C. Elliott. Experimental evaluation of wireless simulation assumptions. In *Proc. 7th Int'l symposium on Modeling, analysis and simulation of wireless and mobile systems (MSWiM 2004)*, pages 78–82, 2004.

[23] W. B. Langdon, M. Harman, and Y. Jia. Efficient multi-objective higher order mutation testing with genetic programming. *Journal of Systems and Software*, 83:2416–2430, 2010.

[24] A. Leitner, M. Oriol, A. Zeller, I. Ciupa, and B. Meyer. Efficient unit test case minimization. In *Proc. 22nd Int'l Conference on Automated Software Engineering (ASE 2007)*, pages 417–420, 2007.

[25] P. Levis, N. Lee, M. Welsh, and D. Culler. TOSSIM: Accurate and scalable simulation of entire TinyOS applications. In *Proc. 1st Int'l Conference on Embedded networked sensor systems (SenSys 2003)*, pages 126–137, 2003.

[26] P. Levis, N. Patel, D. Culler, and S. Shenker. Trickle: a self-regulating algorithm for code propagation and maintenance in wireless sensor networks. In *Proc. 1st Symposium on Networked Systems Design and Implementation - Volume 1*, 2004.

[27] P. Li and J. Regehr. T-check: bug finding for sensor networks. In *Proc. 9th ACM/IEEE Int'l Conference on Information Processing in Sensor Networks*, IPSN '10, pages 174–185, 2010.

[28] Z. Li, M. Harman, and R. M. Hierons. Search algorithms for regression test case prioritization. *IEEE Transactions on Software Engineering*, 33:225–237, 2007.

[29] P. McMinn. Search-based software test data generation: a survey: Research articles. *Softw. Test. Verif. Reliab.*, 14:105–156, 2004.

[30] L. Mottola, T. Voigt, F. Osterlind, J. Eriksson, L. Baresi, and C. Ghezzi. Anquiro: Enabling efficient static verification of sensor network software. In *Proc. 1st Int'l Workshop on Software Engineering for Sensor Networks (SESENA 2010)*, 2010.

[31] C. D. Nguyen, A. Perini, P. Tonella, and F. B. Kessler. Constraint-based evolutionary testing of autonomous distributed systems. In *Proc. 2008 IEEE Int'l Conference on Software Testing Verification and Validation Workshop*, pages 221–230, 2008.

[32] M. Penrose. *Random Geometric Graphs*. Oxford Studies in Probability. Oxford University Press, 2003.

[33] R. Sasnauskas, O. Landsiedel, M. H. Alizai, C. Weise, S. Kowalewski, and K. Wehrle. Kleenet: discovering insidious interaction bugs in wireless sensor networks before deployment. In *Proc. 9th ACM/IEEE Int'l Conference on Information Processing in Sensor Networks*, IPSN '10, pages 186–196, 2010.

[34] L. Tang, Y. Sun, O. Gurewitz, and D. B. Johnson. EM-MAC: a dynamic multichannel energy-efficient mac protocol for wireless sensor networks. In *Proc. 12th ACM Int'l Symposium on Mobile Ad Hoc Networking and Computing*, MobiHoc '11, pages 23:1–23:11, 2011.

[35] M. Woehrle, D. Brockhoff, T. Hohm, and S. Bleuler. Investigating coverage and connectivity trade-offs in wireless sensor networks: The benefits of MOEAs. In *Proc. 19th Int'l Conference on Multiple Criteria Decision Making (MCDM 2008)*, pages 211–221, 2008.

[36] M. Woehrle, K. Lampka, and L. Thiele. Exploiting timed automata for conformance testing of power measurements. In *7th Int'l Conference on Formal Modelling and Analysis of Timed Systems (FORMATS 2009)*, pages 275–290, 2009.

[37] S. Xanthakis, C. Ellis, C. Skourlas, A. L. Gall, S. Katsikas, and K. Karapoulios. Application of genetic algorithms to software testing. In *Proc. 5th Int'l Conference on Software Engineering and its Applications*, pages 625–636, 1992.

[38] T. Yano, E. Martins, and F. L. de Sousa. Most: A multi-objective search-based testing from efsm. In *Proc. 4th Int'l Conference on Software Testing, Verification and Validation Workshops (ICSTW 2011)*, pages 164–173, 2011.

[39] A. Zeller and R. Hildebrandt. Simplifying and isolating failure-inducing input. *IEEE Trans. Softw. Eng.*, 28:183–200, 2002.

[40] E. Zitzler, D. Brockhoff, and L. Thiele. The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicators Via Weighted Integration. In *Proc. Int'l Conference on Evolutionary multi-criterion optimization (EMO 2007)*, pages 862–876, 2007.