Electronic Communications of the EASST
Volume 68 (2014)

Proceedings of the 8th International Workshop on Graph-Based Tools (GraBaTs 2014)

Rapid Prototyping of Topology Control Algorithms by Graph Transformation

Géza Kulcsár, Michael Stein, Immanuel Schweizer, Gergely Varró, Max Mühlhäuser and Andy Schürr

14 pages

Guest Editors: Matthias Tichy, Bernhard Westfechtel
Managing Editors: Tiziana Margaria, Julia Padberg, Gabriele Taentzer
ECEASST Home Page: http://www.easst.org/eceasst/ ISSN 1863-2122

Rapid Prototyping of Topology Control Algorithms by Graph Transformation

Géza Kulcsár[1], Michael Stein[2], Immanuel Schweizer[3], Gergely Varró[4], Max Mühlhäuser[5] and Andy Schürr[6]

[1] geza.kulcsar@es.tu-darmstadt.de, [4] gergely.varro@es.tu-darmstadt.de
[6] andy.schuerr@es.tu-darmstadt.de
Real-Time Systems Lab
Technische Universität Darmstadt, Germany

[2] michael.stein@tk.informatik.tu-darmstadt.de, [3] schweizer@tk.informatik.tu-darmstadt.de
[5] max@tk.informatik.tu-darmstadt.de
Telecooperation Group
Technische Universität Darmstadt, Germany

Abstract: Topology control algorithms are used to improve the energy efficiency (or other quality parameters) of wireless sensor networks. In this paper, we report on the application of graph transformation in the domain of wireless sensor networks by proposing a model-driven rapid prototyping approach for the kTC topology control algorithm to enable the fast implementation and the evaluation of its different variants, and consequently, to accelerate the network quality experimentation cycle. In our approach, wireless sensor networks are described by graph-based models, and three variants of the kTC topology control algorithm are implemented by graph transformation, which are then executed on input network descriptions to derive modified topologies whose quality is then measured in several contexts to be able to assess the achieved network quality improvement.

Keywords: wireless sensor networks, topology control algorithm, rapid prototyping, graph transformation

1 Introduction

In wireless sensor networks (WSNs), battery-powered nodes are employed to sense their environment. Each node is equipped with a wireless transceiver providing easy connectivity. Using these wireless transceivers, nodes transmit and forward sensed data to a common base station.

Unfortunately, wireless communication requires considerable amounts of energy. Hence, energy-efficient communication mechanisms are a major research area in WSNs. Topology control is one such mechanism. Through the removal of network links, each node can reduce its wireless range and save energy. In real deployments, removal decisions can only be taken from the local perspective of a single node without global coordination. Depending on the application, this might lead to crucial links being removed, while unused links are retained, hampering the performance of the overall network. The challenge of application-dependent topology optimization has not been approached yet, as the effort involved in manually evaluating different

combinations of possible application restrictions and metrics has been considered too high.

In order to tackle these kinds of challenges and to achieve much faster prototyping, we study the application of rule-based model transformation techniques (i.e., graph transformation) in the domain of wireless sensor networks.

In this paper, we use a model-driven rapid prototyping approach (i) to consider an application-dependent communication overlay, and (ii) to accelerate the quality assessment process itself. As a first application of this combined approach, we describe wireless sensor networks by graph-based models, and implement the kTC topology control algorithm [SWB+12] using graph transformation. Due to the very simple graph patterns manipulated through kTC this is straightforward. However, it already illustrates the benefits of using graph transformations. With minor effort, two modifications (`LOCAL` and `GLOBAL`) of kTC are implemented on application-specific overlays and evaluated through network simulation. The execution of both `LOCAL` and `GLOBAL` leads to more energy-efficient topologies.

The rest of the paper is structured as follows: Section 2 introduces the domain of wireless sensor networks. Related work is discussed in Sec. 3. Section 4 presents the basics of modeling and graph transformation, while Sec. 5 describes how different variants of the kTC algorithm are implemented by graph transformation. Section 6 quantitatively assesses the network quality improvements achieved by our approach, and Sec. 7 concludes our paper.

2 Background

2.1 Wireless Sensor Networks

Wireless sensor networks (WSNs) have been derived from the idea of smart dust as introduced by Kahn et al. [KKP99]. Smart dust describes the idea of using small (less than one millimeter) and disposable battery-powered nodes with processing, sensing, and wireless transmission capabilities to monitor remote locations.

While the original vision has not yet come to fruition, WSNs are widely employed to provide one of four possible application classes [IKMR08]: (i) event detection and reporting, (ii) data gathering and periodic reporting, (iii) sink-initiated querying, or (iv) tracking-based applications. To provide any application the nodes form a mesh network using their wireless radio. Data from each sensor node is then transmitted to and collected at a common base station, using multi-hop communication. The nodes are powered by batteries, hence, they are infrastructure-independent and can be easily employed even in harsh environments [THGT07].

But the battery puts an upper limit on the node lifetime. Hence, energy efficiency, especially energy efficient communication, is a major research direction of the WSN community. Topology control is one example of such a protocol.

2.2 Topology Control

In WSNs, the nodes form a wireless mesh to transmit data to a common base station. Usually, all nodes use their maximum transmission power $t_{s_{max}}$ to reach a maximum number of nodes. Employing topology control, each node will try to reduce the number of neighbors such that the resulting topology graph will still be connected and retains some other desirable properties. This

edge reduction will allow some or all nodes to reduce their transmission power, hence, saving energy. Unfortunately, reducing the transmission power usually leads to longer routing paths. Most protocols revolve around the balance between aggressive edge reduction and minimizing the increase in path length.

This increase in path length is measured using factors such as hop, length or power stretch. *Hop stretch factor* is defined as the maximum increase in hop distance over all node pairs. The other stretch factors are defined similarly for length and power, respectively. Topologies are called *spanner*, if their stretch factor can be bounded by a given constant. To calculate the increase, topology control assumes a given input topology. In literature most protocols will assume a *unit disk graph (UDG)*. A unit disk graph (UDG) is an undirected graph formed from a collection of points in the Euclidean plane, in which two points are connected by an edge if their distance is below a fixed threshold.

In wireless sensor networks this fixed threshold is the maximum wireless range of the radio assuming homogeneous hardware and no interference.

Topology control is application-independent. Hence, any approach might spare unused links and remove highly important links.

2.3 Data Collection Overlay

The challenge this paper addresses hinges on the following observation. Topology control is employed on the *underlay* only. An underlay U is an undirected graph formed by a collection of nodes, in which two nodes are connected by an *edge* if there is a possible physical link between the nodes. Edges have an application-specific weight parameter called *length*.

Let us assume the most common WSN application, where data is collected from all nodes at one common base station. It would be desirable for any topology control algorithm to change the underlay in a way that the application-specific overlay is almost not affected. Let U be an underlay in a WSN, v_i be an arbitrary node and b be a designated node called *base*. If p_i is a path in U from v_i to b, an *overlay* O is an undirected graph formed by a collection of nodes, in which all nodes v_i are connected to b by an *overlay edge* p_i with *weight* w_i, alternatively denoted as $w(p_i)$. The weight w_i is an application-specific parameter, such as hop count, Euclidean distance, etc.

3 Related Work

After introducing wireless sensor networks and topology control in Sec. 2, we summarize the related work for both topology control and applications of graph transformations on networks.

Most topology control approaches describe *geometrical structures* that can be roughly categorized into using location [LSW05], angle-of-arrival [WLBW01], or being *location-free* using, e.g., link properties as edge weights [LP06, WZ04]. A comprehensive survey of topology control schemes can be found in [Wan08].

With the rise of WSNs, topology control has been of great theoretical interest with the use of various geometrical structures. Geometrical structures like LSθGG [LSW05] provably retain almost all important theoretical properties. Unfortunately, the assumptions made for purely geo-

metrical structures do not hold in real-world deployments. Lately, new algorithms have been proposed to enable topology control in real-world deployments. XTC [WZ04] is a simple, location-free, local topology control algorithm. By design, XTC is much more suitable for real-world deployments. Unfortunately, Lillis et al. [LP06] have shown, that small errors in the weights of XTC may lead to partitioning of the topology.

The kTC algorithm [SWB+12], which is the basis for the rapid prototyping and modeling presented in this paper, was introduced in 2012. kTC is based on the idea to remove the longest edge in a triangle, if the edge is k times longer than the shortest edge. Here, a triangle is a fully meshed subset of three underlay nodes and "shortest" might refer to different lengths, e.g., Euclidean distance, link quality, etc. Due to space restrictions, the interested reader is referred to [SWB+12] for more details on how kTC works. Overall, this simple, local rule provides already up to 17% in energy savings in real-world deployments. These energy savings depend on the application and the communication pattern. Today, these communication patterns or application-specific communication overlays are not reflected in any topology control algorithms due to the complexity of the optimization space.

Network topology problems have been handled by graph transformation in various approaches. In [Gru08], a network of mobile nodes is modeled via graph transformation; however, in contrast to the current technique, this approach aims at reliability through maintaining triangles instead of eliminating them for a gain in energy efficiency.

A meta-model-based graph transformation framework for mobile network analysis has been proposed in [GH04b]. Further similar approaches deal with capturing the dynamic behaviour of (mobile) networks using graph transformation techniques. An application of such a framework to improve mobile network quality through adaptive software architecture is presented in [GH04a]. Although graph transformation is applied here to adapt the software architecture of network devices for maintaining or achieving different quality metrics (instead of adapting the topology itself), both [GH04a] and our approach benefit from a layered view on networks.

[TGM00] takes a more general approach when perceiving distributed systems as network graphs and applies (distributed) graph transformation to describe their changes. [KG12] approaches the challenge of non-deterministic WSN behaviour by introducing probabilistic graph transformation systems. The goal of this work is a realistic transformation-based simulation mechanism and not a static evaluation layer built on an existing simulation framework like our proposed technique. Despite the similar motivation, none of [GH04b, GH04a, TGM00, KG12] address application-independent topology control in the sense it is conceived in this paper.

4 Modeling and Graph Transformation Concepts

This section first gives an overview on basic modeling concepts that are exemplified on underlays and overlays. Afterwards, the technique of *graph transformation* (GT) is shortly introduced.

4.1 Modeling Concepts

A *meta-model MM* is a graph representing concepts of a given domain and capturing the common properties of different system instances within that domain. The nodes of a meta-model

are called *classes* and its edges (capturing the semantic relations between classes) are called *references*. Classes may also have *attributes*. References can be unidirectional or bidirectional. *Multiplicities*, i.e., numeric intervals can be assigned to references, which define a lower and upper bound on how many times a reference can occur in an actual system. A *model M* describes a concrete system instance of a meta-model *MM*.

Example The meta-model for the underlay can be seen in Figure 1.

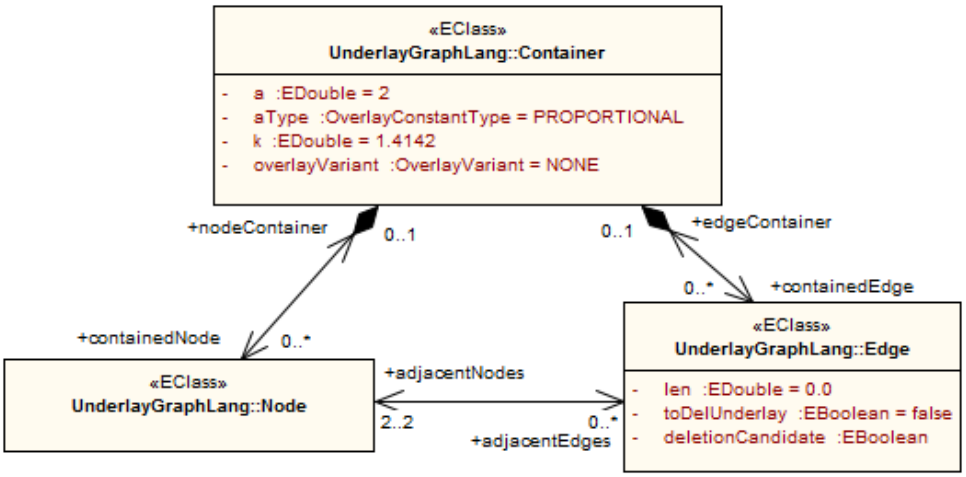


Figure 1: Meta-model for underlay

Network topology nodes and edges are represented by `Node` and `Edge` classes with an adjacency reference between them corresponding to undirected graphs. The elements are additionally contained in a common `Container`.

The meta-model for the overlay and its connections to the underlay is presented in Figure 2.

An overlay has a `BaseNode` and (non-base) `OverlayNodes`. An overlay edge always connects an overlay node to a base node. The overlay element corresponding to an underlay node can be either an `OverlayNode` or a `BaseNode`. All elements are contained in an `OverlayContainer` that determines the application-specific overlay edge weight metric. This container has to contain exactly one base node.

Example Figure 3 presents an underlay U_0 and an overlay O_0, used as a running example throughout the paper. Underlay and overlay edges are depicted as solid and dashed lines, respectively. Circles represent both underlay and corresponding overlay nodes. The numbers on the underlay edges denote lengths, while the bold numbers on overlay edges denote weights. The sum of the edge lengths along the shortest path to the base is used as application-specific weight metric.

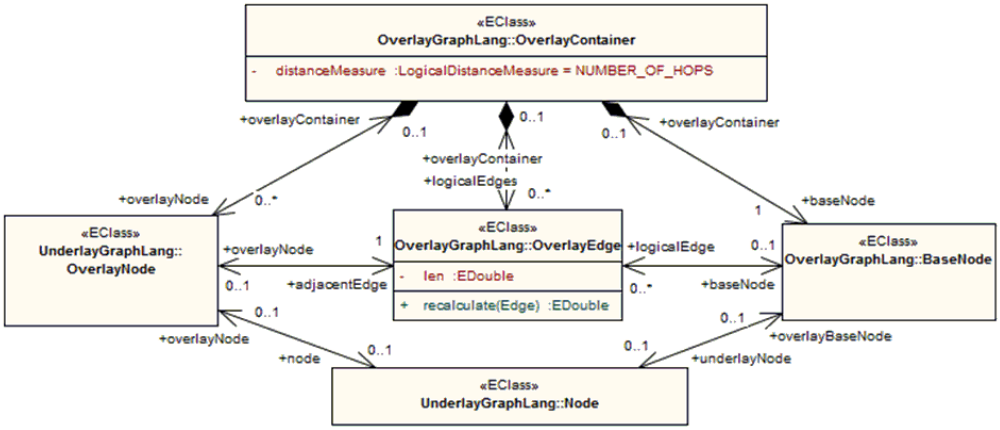


Figure 2: Meta-model for overlay

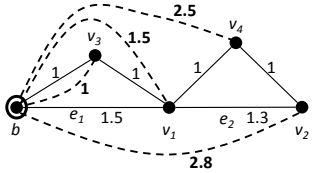


Figure 3: Example underlay and overlay topology

4.2 Graph Transformation

Graph transformation is a rule-based approach to modify graph-based models. A *graph transformation rule r* consists of a left-hand side graph (*LHS*) and a right-hand side graph (*RHS*). An *application* of a rule $r(LHS, RHS)$ to a model M replaces a *match* m of *LHS* in M (i.e., m is a subgraph of M that is homomorphic to *LHS*) with an image of the *RHS*. This is performed by (i) finding a match of *LHS* in M, (ii) removing a part of M that can be mapped to *LHS* but not to *RHS*, and (iii) gluing an image of *RHS* to M by adding new elements that can be mapped to *RHS* but not to *LHS*, yielding the model M'.

Example In this paper, a compact visual syntax is used for rules (instead of giving *LHS* and *RHS*). Black elements are preserved during rule application, while red ones (also marked with '−−') are deleted. An object can have constraints on its attributes: ':=' means value assignment and '==' means equality check.

Figure 4 shows sample graph transformation rules implemented using Story Driven Modeling (SDM) [FNTZ98]. In SDM, a sequence of rule applications is guided by an explicit *control flow*, an additional control structure representing temporal ordering of rules to be applied. Each control flow node contains one rule. For a single-bordered control flow node, the contained rule is applied only *once* (even when there are more possible matches), while a double-bordered control flow node indicates a `for all`-style loop where the contained GT rule is applied on *all* matches.

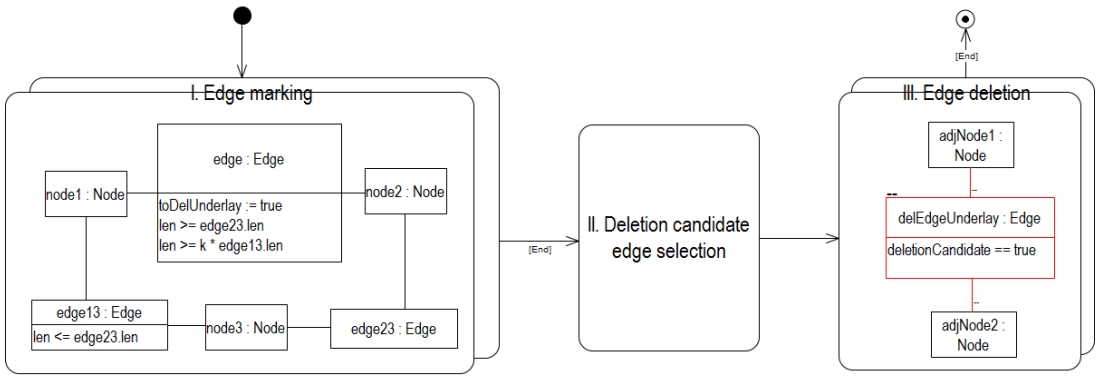


Figure 4: Overview of kTC implementation with GT rules

5 Implementation

The kTC algorithm [SWB+12], which deletes the longest edge in a triangle if it is at least k times longer than the shortest edge in the triangle, has been implemented using eMoflon [Mof], a multi-purpose graph transformation tool. Figure 4 gives an overview of the overall transformation process, given an underlay model U and an overlay model O.

Phase I: Edge marking. In each underlay triangle, the longest edge is *marked* (by setting the `toDelUnderlay` flag) if it is at least k times longer than the shortest edge in the triangle, as expressed by a graph pattern (Fig. 4, on the left).

Phase II: Deletion candidate edge selection. Marked edges are filtered by selecting *deletion candidate edges* according to three different algorithm variants (`NONE`, `LOCAL`, `GLOBAL`), which have been implemented as graph transformation as presented in Sec. 5.1–5.3, respectively (Fig. 4, in the middle).

Phase III: Edge deletion. Finally, deletion candidate edges are actually deleted from the underlay (Fig. 4, on the right).

In the following, E^m and E^{dc} denote the sets of marked edges and deletion candidate edges, respectively.

5.1 Selection of deletion candidates: Variant 1 - NONE

In variant `NONE`, *all* marked edges are considered as deletion candidate edges ($E^{dc} = E^m$), which are then deleted in Phase III.

Example The execution of `NONE` variant is presented in Fig. 5. Consider the initial underlay topology U_0 depicted in Figure 3 with base b and $k = 1.2$.

Let us suppose that Phase I marks edges e_1 and e_2 as the length of the shortest edge is 1 in both triangles, and both e_1 and e_2 are longer than $k * 1 = 1.2$. In Phase II, both e_1 and e_2 are selected as deletion candidate edges. After Phase III, the topology on the right side of Fig. 5 is produced.

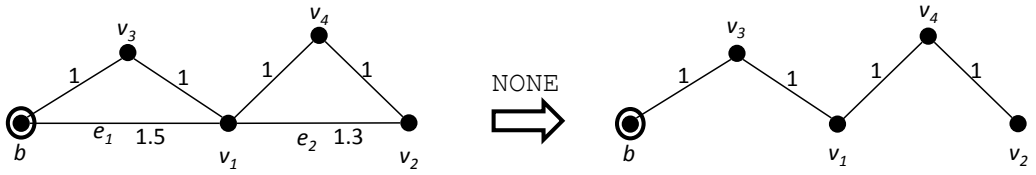


Figure 5: Example for NONE variant

Table 1: Algorithms of LOCAL and GLOBAL variants

Algorithm 1 LOCAL variant

1: $E^{dc} := \emptyset$
2: **for each** $e \in E^m$ **do**
3: $O' := recalc(U, \{e\}, O)$
4: **if** $\mathrm{oc}(e, O, O', a)$ **then**
5: $E^{dc} := E^{dc} \cup \{e\}$
6: **end if**
7: **end for**

Algorithm 2 GLOBAL variant

1: $E^{dc} := \emptyset$
2: $O^* := recalc(U, E^m, O)$
3: **for each** $e \in E^m$ **do**
4: **if** $\mathrm{oc}(e, O, O^*, a)$ **then**
5: $E^{dc} := E^{dc} \cup \{e\}$
6: **end if**
7: **end for**

5.2 Selection of deletion candidates: Variant 2 - LOCAL

In variant LOCAL (Alg. 1), different modified overlays are created for each marked edge. For each marked edge e (line 2), a new overlay O' is created from O by *recalculating* the overlay edges that are adjacent to e (line 3). If the *overlay condition* (line 4) is fulfilled (for e, O, O' and a), then e is added to E^{dc} (line 5).

The *overlay recalculation* ($recalc(U, U^{rec}, O)$) produces a new overlay O' (using underlay edge set U^{rec} in U), having the same node and edge set as O. New weights are calculated as follows:

$$w'(v_i, b) = \begin{cases} \text{length of shortest path from } v_i \text{ to } b \text{ in } U \setminus U^{rec} & \text{if } \exists (v_i, x) \in U^{rec} \\ w(v_i, b) & \text{otherwise} \end{cases}$$

The *overlay condition* ($oc(e, O, O', a)$) for a marked underlay edge $e = (v_1, v_2)$, two overlays O and O' having the same node and edge set, and a constant a is fulfilled if *weight increase conditions* $r(w_1, w_1', a)$ and $r(w_2, w_2', a)$ both hold, where w_1 and w_2 are the weights of overlay edges $p_1 = (v_1, b)$ and $p_2 = (v_2, b)$ in overlay O, respectively, while w_1' and w_2' are the weights of the same overlay edges p_1 and p_2 in overlay O'. The weight increase condition $r(w, w', a)$ expresses that either the ratio or the difference of overlay edge weights w and w' can be at most as large as constant a. In the following, always the ratio of weights is considered.

Figure 6 shows the SDM implementation of the LOCAL variant.

Example Figure 7 presents the execution of LOCAL variant with base b and $a = 1.4$. The initial underlay is again U_0 (Fig. 3). Note that Fig. 7 shows only those overlay edges and weights that

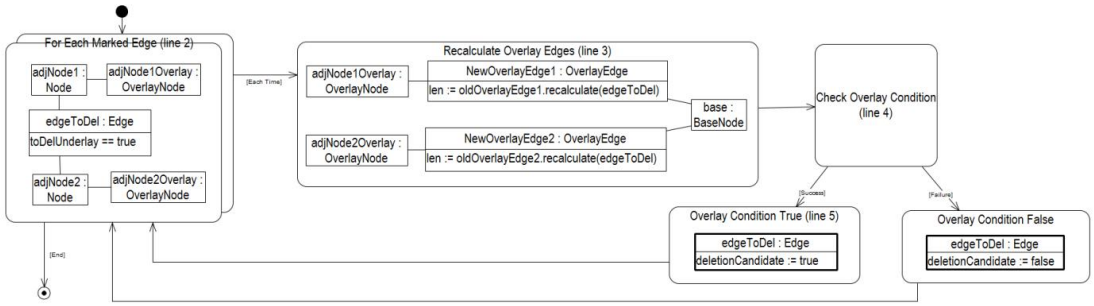


Figure 6: SDM implementation of LOCAL variant

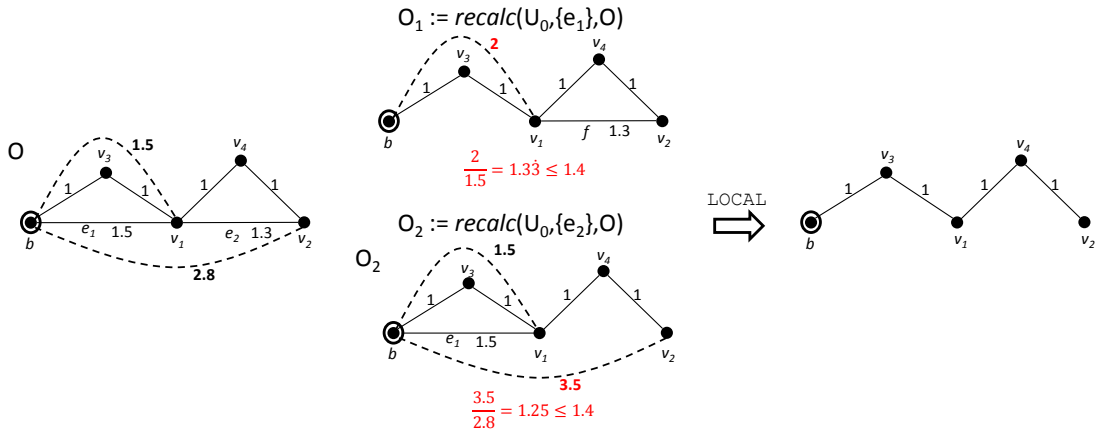


Figure 7: Example for LOCAL variant

are actually recalculated according to the first alternative of the weight recalculation rule applied during overlay recalculation.

Edges e_1 and e_2 are marked in the beginning. First, e_1 is checked. O_1 is created according to the overlay recalculation procedure (see Alg. 1 line 3). Only overlay edge (v_1, b) has to be recalculated as (b, b) would be a loop (when the weight increase condition is considered as fulfilled). Recalculation is done over U_0 without e_1 (see Fig. 7, upper middle). New weight is 2 (along the shortest path $v_1 - v_3 - b$ in the underlay) and as the weight increase condition is true for these values ($\frac{2}{1.5} \leq 1.4$), edge e_1 is selected as a deletion candidate.

As a next step, edge e_2 is checked. While creating O_2 by recalculating the edge weights of O, both (v_1, b) and (v_2, b) have to be recalculated over U_0 without e_2 (see Fig. 7, lower middle). In case of (v_1, b) there is no change so overlay condition is trivially met. In case of (v_2, b), new weight is 3.5 (along the shortest path $v_2 - v_4 - v_1 - b$) and as weight increase condition is true for these values ($\frac{3.5}{2.8} \leq 1.4$), e_2 is also selected as a deletion candidate. After Phase III, the topology

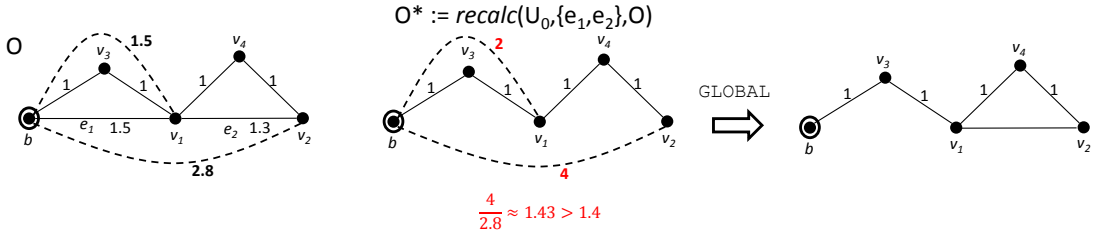


Figure 8: Example for GLOBAL variant

on the right side of Fig. 7 is given as output of the kTC algorithm.

5.3 Selection of deletion candidates: Variant 3 - GLOBAL

In variant GLOBAL (presented in Algorithm 2), a single modified overlay O^* is created, in which weights of those overlay edges are updated over a single modified underlay $U \setminus E^m$, whose non-base end nodes are adjacent to a marked edge (line 2). A marked edge e is selected as a deletion candidate if overlay condition (for e, O, O^* and a) holds (lines 3–7).

Example Figure 8 presents the execution of GLOBAL variant with base b and $a = 1.4$ on the underlay U_0 originating from Fig. 3. Again, only those overlay edges and weights are shown that are going to be recalculated during the process.

Edges e_1 and e_2 are marked in the beginning. In contrast to the LOCAL variant, O^* is created first over U_0 without the marked edges (see Fig. 8, in the middle), and all recalculations are done in this step. As $e_1 = (v_1, b)$ is marked, overlay edge (v_1, b) has to be recalculated ((b, b) not as it is a loop). As $e_2 = (v_1, v_2)$, overlay edges (v_1, b) and (v_2, b) have to be recalculated, so two overlay edges are recalculated altogether: (v_1, b) and (v_2, b). The new weight is 2 for (v_1, b) and 4 for (v_2, b) (along the shortest path $v_2 - v_4 - v_1 - v_3 - b$ in the underlay).

Iterating over the marked edges does not occur until this point. First, e_1 is checked. As for (v_1, b), weight increase condition is true ($\frac{2}{1.5} \leq 1.4$), e_1 is selected as deletion candidate. Afterwards, e_2 is checked. In case of (v_2, b) the weight increase condition is not fulfilled ($\frac{4}{2.8} \not\leq 1.4$), therefore, e_2 is not selected as a deletion candidate edge. After Phase III, the topology on the right side of Fig. 8 is the output of the kTC algorithm.

6 Measurement Results

A simulation study is presented in the following. First, the simulation setup is described. Afterwards, the topologies constructed by the different approaches are examined regarding different metrics from the topology control community.

6.1 Simulation Setup

The simulations were conducted with 500 nodes, including one base station. Both the sensors and the base station are placed uniformly at random onto a squared plane with a side length of 1500 meters. Each node has a maximum transmission range of 157 meters, forming a unit disc graph (UDG).

Three variants of kTC (NONE, LOCAL, and GLOBAL) are considered and compared to the UDG. Two overlay edge weight metrics were applied for evaluation purposes: the sum of the edge length (*edge length metric*) and the number of hops along the shortest path to the base (*hop count metric*). Both the constant a and the overlay edge weight metric were varied for LOCAL and GLOBAL. kTC was always executed with $k = \sqrt{2}$, as this value performs well when aiming at a power-efficient topology used for unicast traffic. Each simulation setup was executed 100 times with different random seeds and all results are given with 95% confidence intervals.

6.2 Results

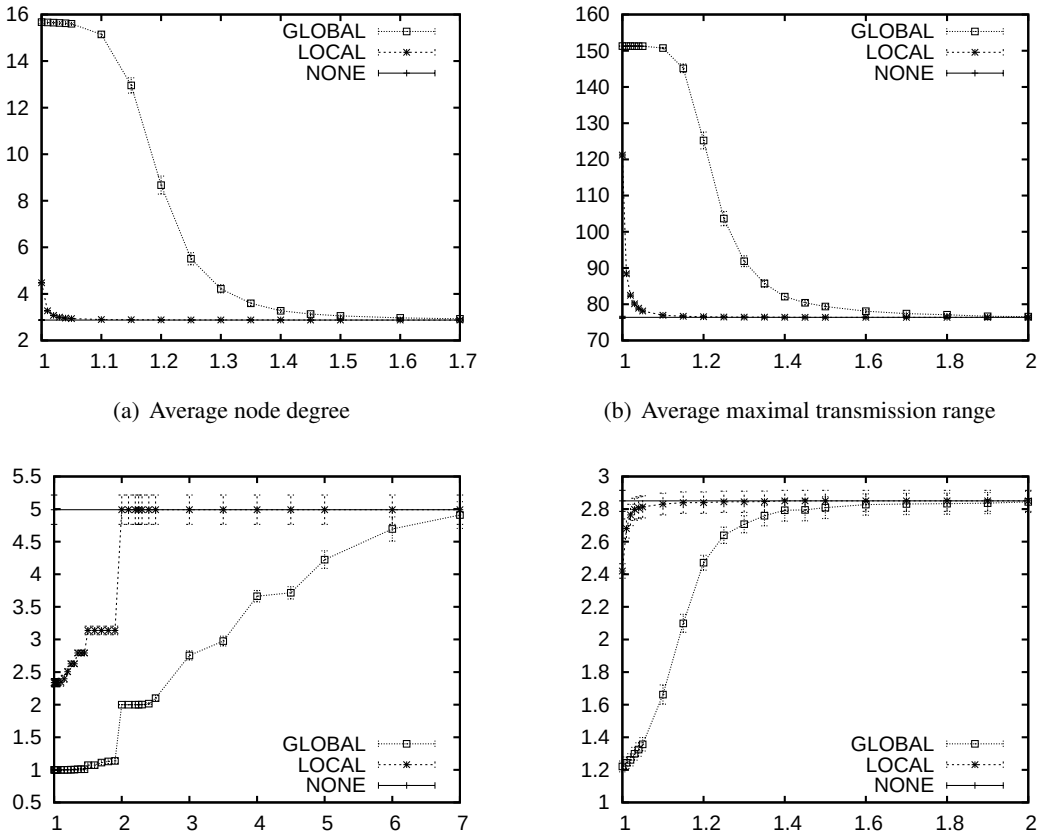
A trivial consequence of the three algorithm variant definitions of Sec. 5 is that the unmodified kTC (variant NONE) removes edges very aggressively aiming for the highest reduction in transmission power, while GLOBAL and LOCAL delete less edges to preserve application-critical edges. This has two important theoretical implications. First, both LOCAL and GLOBAL are connected, if NONE is connected, and the connectivity of variant NONE has been proven for the case, when the input topology is connected [SWB+12]. Second, while NONE has proven planarity if $k \leq \sqrt{2}$, this is obviously not true for both LOCAL and GLOBAL.

This result transfers to other important metrics. We expect the node degree and transmission range for LOCAL and GLOBAL to be higher. Figures 9(a) and 9(b) show the average node degree and maximal transmission range for all three variants. Both look almost identical and, as expected, LOCAL and GLOBAL start off higher than NONE. Both converge against NONE as a is increased and less edges are retained. It also confirms that for small a LOCAL removes edges more aggressively than GLOBAL. Topology control is about shrinking the transmission range and the number of neighbors, hence, those results at first glance indicate a decrease in performance.

However, the network is supposed to route data to the base station. Hence, the stretch factor to the base station is crucial to the overall performance. Here, stretch factor is the maximum increase in path length (either measured in hops or length) to the base station. Figures 9(c) and 9(d) show hop and length stretch, respectively. We can observe that NONE has a length stretch of 2.8 and a hop stretch of around 5. This indicates that, while energy is saved from reduced transmission ranges, energy is also wasted due to an aggressive increase in hop count and length. For $a = 1$, GLOBAL has a hop stretch of almost 1, while LOCAL has around 2.5. Again, we can observe that both converge against NONE as a is increased.[1]

As a summary, the execution of both LOCAL and GLOBAL leads to topologies with smaller stretch factors than variant NONE. However, this improvement comes with a price. There is a trade-off with respect to the number of removed edges. On the one hand, removing many edges (variant NONE) results in a planar topology with bounded node degree and a low average

[1] Note: The jumps of the hop stretch factor are due to the fact that integer increases of a will allow edges closer to the base station to be removed.



(a) Average node degree

(b) Average maximal transmission range

(c) Hop stretch factor

(d) Length stretch factor

Figure 9: Simulation results for `NONE`, `LOCAL` and `GLOBAL` (x-axis: a)

maximum transmission range. On the other hand, keeping more edges decreases the stretch factors. Overall, we argue that the trade-off presented by `LOCAL` offers both, much lower stretch and almost no increase in node degree and transmission range. In the future, we will investigate porting `LOCAL` back into hardware to evaluate real applications.

7 Conclusion

In this paper, we proposed a model-driven rapid prototyping approach to enable the fast implementation and evaluation of three variants of the kTC topology control algorithm, which have been realized by graph transformation. Additionally, the achieved network quality improvement has been assessed by executing the three algorithm variants on input network descriptions to derive modified topologies whose quality is then measured in several contexts.

Our proposed technique currently relies on static and global snapshots of wireless sensor network topologies, although in real-world ad-hoc network scenarios, (i) nodes and edges can join

or leave the topology dynamically, and (ii) they should manage themselves based on local information. These requirements could only be satisfied, if graph pattern matching engines supported an incremental and distributed setup, which is definitely not the case by the state-of-the-art tools.

As part of the Collaborative Research Center MAKI [MAK], we are currently integrating an event-based network simulator (PeerfactSim.KOM) and a graph transformation tool (EMoflon). This work is the first major result from this ongoing integration. We envision major contributions to both network and graph transformation research. The graph patterns manipulated through kTC are relatively simple and we plan to use more complex patterns (e.g., motifs [KSBS10]) in the future. We also envision more dynamic environments in the future, where node mobility, link properties, etc. require iterative approaches.

For kTC, our future activities involve the extension of the proposed rapid prototyping approach by (i) the implementation and evaluation of further algorithm variants, (ii) additional quantitative experiments, in which a large combination of metrics and measurement configurations is used, and (iii) implementing the most promising approaches in hardware for further evaluation.

Acknowledgements: Funded by the DFG as part of the CRC 1053 MAKI.

Bibliography

[FNTZ98] T. Fischer, J. Niere, L. Torunski, A. Zündorf. Story Diagrams: A New Graph Rewrite Language based on the Unified Modeling Language. In Engels and Rozenberg (eds.), *Proc. of the 6th International Workshop on Theory and Application of Graph Transformation*. LNCS 1764, pp. 296–309. Springer, 1998.

[GH04a] P. Guo, R. Heckel. Conceptual Modeling of Styles For Mobile Systems: A Layered Approach Based on Graph Transformation. In Lawrence et al. (eds.), *Proc. of the Conference on Mobile Information Systems*. International Federation for Information Processing 158, pp. 65–79. Springer, 2004.

[GH04b] P. Guo, R. Heckel. Modeling and Simulation of Context-Aware Mobile Systems. In *Proc. of the 19th IEEE International Conference on Automated Software Engineering*. Pp. 430–433. IEEE Computer Society, 2004.

[Gru08] S. Gruner. Graph Transformation Model of a Triangulated Network of Mobile Units. In Ermel et al. (eds.), *Proceedings of Seventh International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2008)*. Electronic Communications of the EASST 10. Feb. 2008.

[IKMR08] A. Iyer, S. Kulkarni, V. Mhatre, C. Rosenberg. A Taxonomy-based Approach to Design of Large-scale Sensor Networks. In Li et al. (eds.), *Wireless Sensor Networks and Applications*. Signals and Communication Technology, pp. 3–33. Springer, 2008.

[KG12] C. Krause, H. Giese. Probabilistic Graph Transformation Systems. In Ehrig et al. (eds.), *Graph Transformations*. Lecture Notes in Computer Science 7562, pp. 311–325. Springer Berlin Heidelberg, 2012.

[KKP99] J. M. Kahn, R. H. Katz, K. S. J. Pister. Next Century Challenges: Mobile Networking for "Smart Dust". In *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking*. Pp. 271–278. ACM, New York, NY, USA, 1999.

[KSBS10] L. Krumov, I. Schweizer, D. Bradler, T. Strufe. Leveraging network motifs for the adaptation of structured peer-to-peer-networks. In *IEEE Global Telecommunications Conference*. 2010.

[LP06] K. Lillis, S. Pemmaraju. Topology control with limited geometric information. In *Principles of Distributed Systems*. 2006.

[LSW05] X.-Y. Li, W.-Z. Song, W. Wang. A unified energy-efficient topology for unicast and broadcast. In *MobiCom*. 2005.

[MAK] MAKI webpage. http://www.maki.tu-darmstadt.de/sfb_maki/.

[Mof] eMoflon webpage. http://www.emoflon.org/.

[SWB+12] I. Schweizer, M. Wagner, D. Bradler, M. Mühlhäuser, T. Strufe. kTC – Robust and Adaptive Wireless Ad-Hoc Topology Control. In *Proc. of the 21st International Conference on Computer Communications and Networks*. 2012.

[TGM00] G. Taentzer, M. Goedicke, T. Meyer. Dynamic Change Management by Distributed Graph Transformation: Towards Configurable Distributed Systems. In Ehrig et al. (eds.), *Proc. of the 6th Int. Workshop on the Theory and Application of Graph Transformations*. LNCS 1764, pp. 179–193. Springer, Paderborn, Germany, 2000.

[THGT07] I. Talzi, A. Hasler, S. Gruber, C. Tschudin. PermaSense: Investigating Permafrost with a WSN in the Swiss Alps. In *Proceedings of the 4th Workshop on Embedded Networked Sensors*. Pp. 8–12. 2007.

[Wan08] Y. Wang. Topology Control for Wireless Sensor Networks. In *Wireless Sensor Networks and Applications*. 2008.

[WLBW01] R. Wattenhofer, L. Li, P. Bahl, Y. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Infocom*. 2001.

[WZ04] R. Wattenhofer, A. Zollinger. XTC: a practical topology control algorithm for ad-hoc networks. In *Parallel and Distributed Processing Symposium*. 2004.