# a-kTC: Integrating Topology Control into the Stack

Immanuel Schweizer, Ralf Zimmermann, Michael Stein, Max Mühlhäuser
Technische Universität Darmstadt, Telecooperation Lab
Email: schweizer@cs.tu-darmstadt.de

***Abstract*—Topology control for wireless sensor networks is a thriving research field with no real-world adoption. This is partly due to the fact that a large number of approaches is focused on providing theoretical bounds. However, even more recent practical approaches are not being implemented. Integrating topology control with medium access control (MAC) and routing into a standard communication stack seems to be a non-trivial challenge. Some experts even doubt real-world efficiency gains.**

**In this paper, we present asynchronous kTC (a-kTC). It builds upon the kTC topology control approach and integrates the protocol into the standard Contiki communication stack. We report up to $16\%$ reduction in overall energy consumption in the testbed for many-to-one and all-to-all communication. This is the first topology control evaluation indicating real-world reductions in total energy consumption.**

***Index Terms*—topology control, wireless networks, communication stack, testbed**

## I. Introduction

Topology control in wireless networks does – in theory – promise energy savings through a decrease in transmission range. In wireless sensor networks (WSNs) small and battery-constrained nodes coordinate to provide a common service. One would assume ample deployments with and implementations of topology control. However, while, e.g., energy-efficient medium access control (MAC) and routing protocols are standard components, which are added to the relevant operating systems, topology control has remained a largely theoretical endeavor.

Nonetheless, topology control has been a thriving research topic with a multitude of algorithms proposed in the past [1]. Most early approaches to topology control build geometric structures with certain graph properties, like planarity and bounded node degree. However, even though all approaches promise energy-efficient and beneficial topologies, there is no real-world adoption. A reason for this is that most topology control algorithms rely on unrealistic assumptions. Consequently, a considerable amount of researchers even argue that topology control provides no real-world advantages.

This perception has not changed with more practical approaches like XTC [2] and kTC [3]. While most protocols have a proven performance record in simulation [4], the leap into real-world networks has been infeasible so far. To the best of our knowledge there is no study showing any energy savings integrating topology control in the communication stack, i.e., with state of the art routing and medium access control.

This paper presents asynchronous kTC (a-kTC), which builds upon our earlier work, kTC [5]. a-kTC is implemented on Contiki and integrates with ContikiMAC [6] and the Rime communication stack. a-kTC, compared to kTC, further reduces overhead by reducing the amount of broadcasted message types to one, thus enabling piggybacking on any standard discovery protocol. a-kTC also drops the assumption of network synchronization to work with any real-world deployment.

We evaluated a-kTC in the testbed based on two different application scenarios: (i) wireless mesh network with a many-to-many and (ii) wireless sensor networks with a many-to-one communication pattern. Overall, we report average energy savings of up to $16\%$ per node, with a small decrease in packet reception rates. These energy savings are in addition to the already energy-efficient Contiki communication stack. This is the first study able to show significant overall energy saving with topology control integrated into the communication stack.

## II. Asynchronous kTC (a-kTC)

*kTC* [3] is a practical topology control algorithm. It is able to deal with link inconsistencies, works with low overhead and is able to produce stable and efficient topologies. Our simulations in [3] illustrate the advantage of kTC against other state-of-the-art approaches. However, kTC has one main shortcoming: It requires time synchronization for the first two broadcasts.

The time synchronization requirement limits adaptation and integration into existing communication stacks. Hence, this paper contributes asynchronous kTC (a-kTC). a-kTC removes the need for time synchronization while retaining the performance of kTC. Why is synchronization required for kTC? kTC requires a *complete* and *consistent* view of the 2-hop neighborhood for any local decision. With kTC the synchronization is required to ensure a consistent state among all nodes when the local decision (Round 3) is taken. However, with some adaptations we can remove the consistency constraint. If we can then guarantee connectivity and even convergence against the kTC topology over time, we can drop the synchronization requirement and, thus, work in real-world deployments.

In kTC, different messages are exchanged in synchronized rounds to build a consistent view of the 2-hop neighborhood. In a-kTC only one message, the ONEHOPMESSAGE (OHM) is sent periodically. Each node will store information about 1-hop and 2-hop neighbors and their respective edge weights into a neighbor table $N$. A node will execute three methods, which are as follows:

**SEND ONEHOPMESSAGE** Each node $u$ will periodically extract a subset $N_1$ from $N$. This subset contains all 1-hop neighbors of $u$ and their edge weight to $u$. $N_1$ is then broadcasted as a OHM to all 1-hop neighbors. This broadcast

978-1-4673-6770-7/15/$31.00 ©2015 IEEE

is sent using the maximal transmission power setting of a given node, i.e., it will always reach all possible physical neighbors.
**RECEIVE ONEHOPMESSAGE** For each received OHM a node will update the neighbor set $N$. Let $s$ be the sender of the message and $u$ the receiver. $u$ will measure the link quality as edge weight $w(s,u)$, e.g., using the RSSI. If $s \in N$ then $u$ will update the edge weight in $N$ with $w(s,u)$ with an aggregate function (e.g., average, min, max, etc.). Otherwise it will add $s$ and $w(s,u)$ to $N$. All other vertices and edge weights, which are contained in the OHM, are either updated or added accordingly.
**UPDATE TOPOLOGY** Update topology is also executed periodically by each node $u$. The method will copy $N$ into $N'$ and then check $N'$ for triangles. Triangles are three vertices in $N'$, which are fully connected. The longest edge in the triangle is removed from $N'$, if it is adjacent to $u$ and $k$ times longer than the shortest edge. $u$ will then check all direct neighbors in $N'$ for the neighbor $b$ with the highest edge weight (the lowest link quality). $u$ can reduce the transmission power such that $b$ is still connected.

As stated above, we can no longer guarantee that *UPDATE TOPOLOGY* is executed on the complete 2-hop neighborhood. This might impair the performance as some links might not be removed, but it cannot lead to disconnects. a-kTC will remove either the same or less links than kTC. As a-kTC collects more information over time, a-kTC will gradually converge against kTC. We will now formalize this convergence.

*Convergence of a-kTC:* a-kTC applies the same rules as kTC. However, it might do so using an incomplete view of the 2-hop neighborhood. We will now provide some results on convergence of a-kTC to kTC. For this section the following assumptions hold: First, a connected, symmetric, and static[1] input topology $G$ is given. $G$ has $m$ edges and $n$ vertices. Messages are transmitted in a finite time $t < \Delta t$ and are delivered with a probability $p > 0$. kTC is executed once on $G$ with no message loss ($p = 1$). The output is $G_{kTC}$. a-kTC is also executed on $G$. Starting at time $t = 0$, each node executing a-kTC periodically sends ONEHOPMESSAGES (OHM) and updates the local topology. $G_{a-kTC_t}$ is the output topology for time $t \geq 0$.

We observe the following: After all $n$ nodes have received at least one *OHM* per neighbor, each node has a complete view of the 1-hop neighborhood. Let us denote the time all nodes have at least received one *OHM* with $t_1$. If all nodes have received at least another *OHM* from each neighbor after $t_1$, each node is aware of the complete 2-hop neighborhood. As $a_{kTC}$ applies the same rule as $kTC$ and given the static topology assumption, having acquired the complete 2-hop neighborhood results in the same edges being removed.

The following is immediately obvious:

**Lemma 1.** $\forall\, t:\ G_{a-kTC_t} \subseteq G_{kTC}$.

Even under message loss ($0 < p < 1$) on one or more links, we can prove the following result:

**Theorem 1.** $\lim_{t\to\infty}\ G_{a-kTC_t} = G_{kTC}$.

*Proof:* To prove the theorem it suffices to show that their are two distinct times $t_1$ and $t_2$, where all OHM messages have been received. Distinct means that the time difference $t_2 - t_1$ is large enough such that each node can execute SEND ONEHOPMESSAGE at least two times, i.e., send at least two OHM messages. Let us denote the minimum time required for distinction with $\widetilde{t}$.

Let $\widehat{t_1}$ denote the time all nodes have sent their *first* OHM. We can now pick an arbitrary time $t_i > \widehat{t_1} + \Delta t$. The probability for the last $m$ messages (one last message per link) to be received is $p_{all} = p_1 * p_2 * ... * p_m$, where $p_i$ denotes the delivery rate per link. Note that $0 < p_{all} < 1$, since all $p_i > 0$ and at least one $p_i < 1$. Let us pick a second time $t_{i+1} = t_i + \widetilde{t}$. Again, the probability for all $m$ messages to be received is $p_{all}$.

Now, $t_i$ and $t_{i+1}$ are independent identically distributed Bernoulli trials and the outcome is either 1 (all nodes have received the last OHM) or 0 (at least one node has not received the last OHM) with probability $p_{all}$ and $1 - p_{all} = q$ respectively. At any time $t$ we can conduct up to $n$ such trials ($t_i$ to $t_{i+n}$). The total probability P of two or more trials yielding 1 is given as $P = 1 - (1 - p_{all})^n - n * p_{all} * (1 - p_{all})^{n-1}$. Since $\lim_{t\to\infty} n \to \infty$, we now need to prove that $\lim_{n\to\infty}\ P = 1 - q^n - n * p_{all} * q^{n-1} = 1$. Observe that $\lim_{n\to\infty}\ q^n = 0$ and $\lim_{n\to\infty}\ n * p_{all} * q^{n-1} = 0$, thus, the proof is complete. ■

This concludes the proof on convergence. a-kTC allows for an asynchronous execution on each node while converging against kTC. Also, since there is only one message type remaining, this information can be piggybacked onto any given discovery protocol. This further reduces overhead and makes the protocol more agile. The update period can be chosen by each node based on the volatility of its neighborhood.

## III. Evaluation

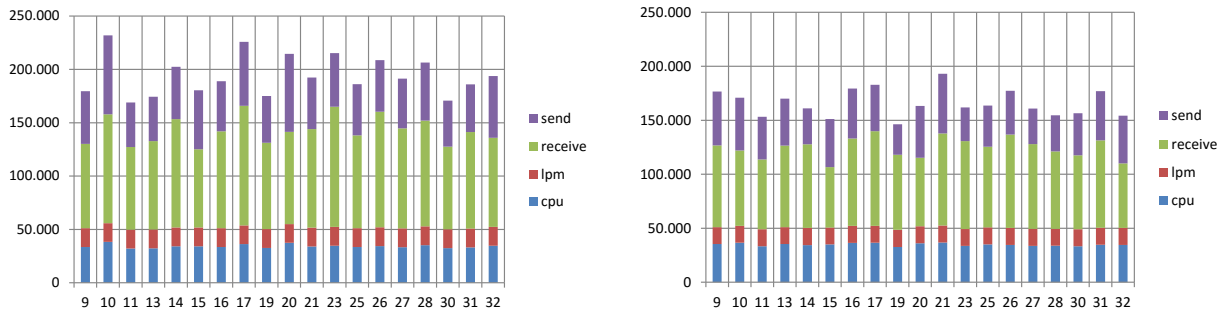
We have implemented a-kTC for the well renowned Contiki operating system[2], and deployed the implementation in a WSN testbed. To evaluate the practicality of a-kTC, we report results for two different scenarios, wireless mesh and wireless sensor networks, with two distinct communication patterns, all-to-all and many-to-one, respectively.

### A. Testbed Setup

We used the Piloty site of the TUD$\mu$Net testbed [7] in Darmstadt to evaluate a-kTC. 20 TelosB-compatible nodes were deployed over 8 rooms on a single floor. The nodes are equipped with a CC2420 radio [8] for 802.15.4 compliant wireless communication. The location of each node is displayed in Figure 1. In the many-to-one scenario, all sampling nodes (hexagonal) are transmitting their data to the base station (triangle).

[1]This includes no node failures and links are not changing with regard to delivery rate and weight as utilized by (a-)kTC

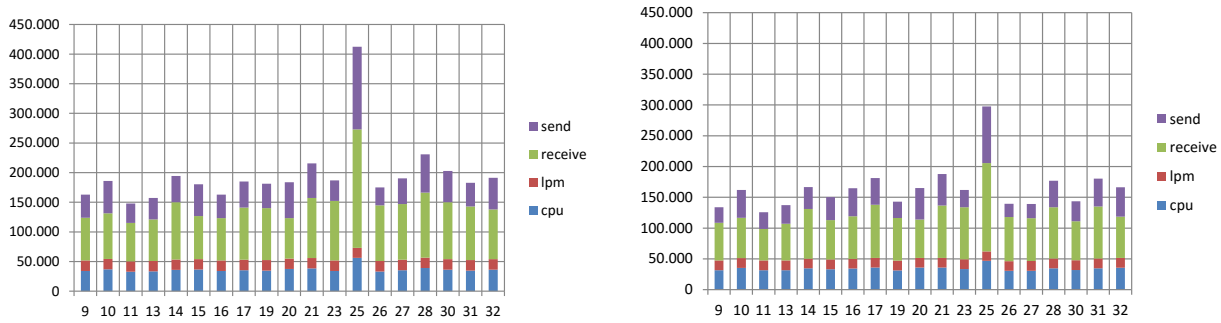
[2]ContikiOS 2.5 was used




(a) Without a-kTC

(b) a-kTC with k = 1.41

Fig. 2: Energy consumption per node (Wireless Mesh Network)




(a) Without a-kTC

(b) a-kTC with k = 1.41

Fig. 3: Energy consumption per node (Wireless Sensor Network)

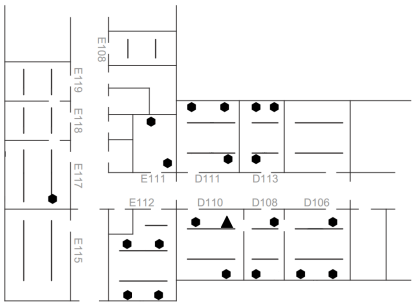



Fig. 1: Placement of nodes in the TUD$\mu$Net testbed

To implement a-kTC, we decided to use Rime [9] as communication stack. Rime is the default communication stack of ContikiOS. We used ContikiMAC [6] as MAC Layer.

In the TUD$\mu$Net testbed each node is connected to a power outlet. Hence, it is not possible to measure energy consumption directly. To this end, Contiki provides the Energest framework [10] to evaluate the energy-efficiency of network protocols. The Energest framework keeps track of the time (in ticks) different system components are used by ContikiOS. Based on this information, we are able to compute the energy consumption for each node individually.

As explained before, topology control saves energy by reducing the transmission range. For our implementation of a-kTC, we use the Received Signal Strength Indicator (RSSI) as edge weight. RSSI offers a good indicator of link quality and can be read from the transceiver immediately after a transmission is received. The challenge is to set the power just high enough to reach the farthest neighbor based on the measured RSSI value. We performed a thorough measurement study to decide on this mapping manually. This also offered a much better insight into the testbed. Next, we report on the result of our deployment for both use cases.

### B. Results

As mentioned before, we considered two scenarios: The first scenario is a *Wireless Mesh Network*, and the second scenario is a *Wireless Sensor Network*. Both scenarios offer distinct communication patterns. In wireless mesh networks, we study all-to-all communication, where each node can possibly transmit to any other node. In wireless sensor networks, we study many-to-one communication, where each node will transmit packets to a common base station. Our evaluation is concerned with the reduction in total energy consumption given topology control. Thus, we measured the total energy consumption per node (in mWs). This includes consumption of the three RF transceiver modes, sending, receiving, and low power listening, and the consumption due to the CPU. We also report the delivery rate to evaluate the effect of reduced transceiver range.

*1) Wireless Mesh Network:* Wireless mesh networks enable communication between arbitrary wireless nodes. They are, for example, used in disaster response if other means of communication fail. Here, a given node might communicate with any other node. Each node will wait a given period of time to choose a receiver uniformly at random. A predefined payload is then sent to this random receiver.

Three runs, each with a duration of $4$ hours, have been conducted. Approximately every $30$ minutes, each node sends the a-kTC message and updates the topology accordingly. The total energy consumption per node is given in Figure 2.

The configuration without a-kTC yields an average energy consumption per node of $194,351.05mWs$. With a-kTC the energy consumption per node is reduced to $169,383.3mWs$ on average. A total reduction of $12.85\%$ over all runs. Since we use a state of the art communication stack these energy savings are additional savings realized by the topology control applied here. The results are summarized in Table I.

| | cpu | lpm | receive | send | total | delivery rate |
|---|---|---|---|---|---|---|
| without a-kTC | 34,196.89 | 17,933.84 | 91,114.74 | 51,405.58 | 194,351.05 | $72,89\%$ |
| a-kTC (k=1.41) (1) | 34,774 | 15,742 | 78,265.26 | 42,132.79 | 170,914.05 | $68,57\%$ |
| a-kTC (k=1.41) (2) | 34,830.63 | 15,718.21 | 74,096.84 | 41,331 | 165,976.68 | $63,15\%$ |
| a-kTC (k=1.41) (3) | 35,067.89 | 15,697.16 | 78,265.26 | 42,132.79 | 170,914.05 | $68,32\%$ |

TABLE I: Metrics as average per user

The reduced transmission range seems to have an only marginal effect on message delivery rates. Without a-kTC, $4,482$ messages out of $6,149$ messages total have been delivered. This results in $72.89\%$ delivery rate. The delivery rates with a-kTC range from $63.15\%$ to $68.57\%$.

In summary, delivery rates are low for this testbed no matter if a-kTC is applied. Further studies with different MAC protocols might yield better rates. The insignificant drop in delivery rates is offset by energy savings of over $12\%$. The savings are additional to the efficient communication stack and are higher than expected.

*2) Wireless Sensor Networks:* In wireless sensor networks, especially in urban environments, periodic reporting is the most frequent application. All nodes report their environmental measurements to a given base station. In this scenario, all nodes will wait a given period of time to send a predefined payload to node $25$. Node $25$ was chosen as base station as it is located at the center of the testbed.

Again, three runs, with a duration of $4$ hours each, have been conducted. Each $30$ minutes, a node runs a-kTC. The total energy consumption per node is given in Figure 3.

Based on this configuration, the run without a-kTC yields an average energy consumption per node of $196,297.58mWs$. With a-kTC the energy consumption per node is reduced to $164,074.7mWs$ on average. As illustrated, node $25$ has a higher energy consumption as it was set as the base station. Overall we obtain a total reduction of $16.42\%$. Again, this is additional to the energy savings already achieved by MAC and routing. The results are summarized in Table II.

In this run the reduced transmission range seems to have a marginally higher effect on message delivery. Without a-kTC, $3,807$ messages out of $5,752$ messages total have been

| | cpu | lpm | receive | send | total | delivery rate |
|---|---|---|---|---|---|---|
| without a-kTC | 36,544.84 | 17,617.95 | 91,509.47 | 50,625.32 | 196,297.58 | $66,19\%$ |
| a-kTC (k=1.41) (1) | 34,981 | 15,732.58 | 76,026.32 | 41,302.16 | 168,033.05 | $56,17\%$ |
| a-kTC (k=1.41) (2) | 34,208.79 | 15,755.53 | 74,848.42 | 39,579.74 | 164,392.5 | $64,58\%$ |
| a-kTC (k=1.41) (3) | 34,088.05 | 15,754.53 | 71,652.63 | 38,303.42 | 159,798.6 | $65,06\%$ |

TABLE II: Metrics as average per user

delivered. This results in $66.19\%$ delivery rate. With a-kTC, the delivery rate ranges from $56.17\%$ to $65.06\%$.

The energy saved in both runs was higher than $12\%$. An exceptional result considering that the nodes already save energy using very energy efficient MAC and routing protocols.

## IV. Conclusion

In summary, this is the first paper to report significant energy savings with applied topology control. This is mainly due to our integration of a-kTC with the standard communication stack, especially the MAC protocol. Given this integration, we could evaluate the total energy consumption for two real-world application scenarios, wireless sensor and wireless mesh networks. We could show energy savings of $12\%$ for all-to-all communication (wireless mesh networks) and up top $16\%$ for many-to-one (wireless sensor networks).

Based upon this work, we want to study a-kTC on different testbeds and integrate it into different communication stacks to further evaluate the effect of topology control, especially on packet reception rate.

## Acknowledgment

This work has been funded by the German Research Foundation (DFG) as part of project A01 within the Collaborative Research Center (CRC) 1053 – MAKI.

## References

[1] Y. Wang, "Topology Control for Wireless Sensor Networks," in *Wireless Sensor Networks and Applications*, ser. Signals and Communication Technology. Springer US, 2008, ch. 5, pp. 113 – 147.

[2] R. Wattenhofer and A. Zollinger, "XTC: a practical topology control algorithm for ad-hoc networks," in *18th International Parallel and Distributed Processing Symposium*, 2004.

[3] I. Schweizer, M. Wagner, D. Bradler, M. Mühlhäuser, and T. Strufe, "kTC - Robust and Adaptive Wireless Ad-hoc Topology Control," in *IEEE ICCCN*, 2012.

[4] F. Fuchs, M. Völker, and D. Wagner, "Simulation-based analysis of topology control algorithms for wireless ad hoc networks," in *Design and Analysis of Algorithms*. Springer, 2012.

[5] I. Schweizer, R. Bärtl, A. Schulz, F. Probst, and M. Mühlhäuser, "NoiseMap - Real-time participatory noise maps," in *Second International Workshop on Sensing Applications on Mobile Phones*, 2011.

[6] A. Dunkels, L. Mottola, N. Tsiftes, F. Österlind, J. Eriksson, and N. Finne, "The announcement layer: Beacon coordination for the sensornet stack," in *Wireless Sensor Networks*, 2011, pp. 211–226.

[7] P. E. Guerrero, A. P. Buchmann, A. Khelil, and K. Van Laerhoven, "TUD$\mu$Net, a Metropolitan-Scale Federation of Wireless Sensor Network Testbeds," in *9th European Conference on Wireless Sensor Networks*, Feb. 2012.

[8] A. S. Chipcon, "CC2420 datasheet."

[9] A. Dunkels, "Rime - a lightweight layered communication stack for sensor networks," in *European Conference on Wireless Sensor Networks (EWSN)*, 2007.

[10] A. Dunkels, F. Österlind, N. Tsiftes, and Z. He, "Software-based on-line energy estimation for sensor nodes," in *Proceedings of the 4th workshop on Embedded networked sensors*, 2007.