S-XTC: A Signal-Strength Based Topology Control Algorithm for Sensor Networks

Matthias Dyer, Jan Beutel and Lothar Thiele
Computer Engineering and Networks Laboratory
ETH Zurich, Switzerland
{dyer, beutel, thiele}@tik.ee.ethz.ch

***Abstract*— We present S-XTC, a topology control algorithm that uses the received signal strength indicator (RSSI) of the radio on wireless sensor nodes. The algorithm is based on XTC, a practical topology control algorithm that constructs a relative neighborhood graph. In contrast to the pure XTC our extensions add to the robustness and resilience against fluctuation in the RSSI values. While guaranteeing connectivity and a bounded node degree, network topologies are able to adapt to gradual changes in the network and it's environment. In this paper, we discuss the shortcomings of the previous algorithm and evaluate S-XTC by analytical proofs and simulation results. Furthermore, we have successfully implemented and tested S-XTC on the BTnode platform of which we discuss a case study and performance evaluation.**

I. Introduction

Many applications for wireless sensor/actuator networks (WSNs) require that all nodes are connected in a common network. Consider for example a sensor network for environmental monitoring, where the nodes regularly report measurement data to a central base station. In other applications, possibly with distributed real-time control such as tracking applications or wireless alarm systems, the continuous connectivity of the network is even more important and often mission critical.

On the other hand, the applications usually run on small devices with very limited resources in terms of computation, communication and energy. In order to maximize the lifetime of the nodes, energy efficient algorithms are required on all layers of an application. One possibility for saving energy is to choose the right network topology. Especially in dense networks, where a node is within the transmission range of many other nodes, selecting an optimized subset of neighbors with which to communicate has a large potential to increase overall network performance and most important to reduce the power consumption.

A topology control algorithm that runs on wireless sensor nodes, selects a set of neighbors for communication, such that the resulting topology has advantageous properties. It can have several objectives. In this work the goal is to reduce the number of communication links and to avoid bad quality connections, while guaranteeing a robust network with redundant connections whenever possible. When G is the visibility graph, containing an edge for every *possible* link between two nodes, and G_{TC} the reduced graph obtained through topology control, with only the *selected* links, the connectivity is guaranteed if G_{TC} is connected, whenever G is connected. This task is challenging, because the algorithm needs to be fully distributed and only local information can be used for the decision-making in order to reduce traffic overhead. Furthermore a topology control algorithm should be simple and practical, a dominant prerequisite for successful implementation on commonly available sensor node platforms, i.e. additional hardware for localization should not be required.

The work presented here was supported by the National Competence Center in Research on Mobile Information and Communication Systems (NCCR-MICS), a center supported by the Swiss National Science Foundation under grant number 5005-67322.

A. Contributions

In this work we present S-XTC, a topology control algorithm that uses the received signal strength indicator (RSSI) of the radio on the sensor nodes as a link-metric. The novel contribution of this paper is twofold: To the best of our knowledge, the presented algorithm is the first one that combines (i) energy efficiency through topology control based on a single link-metric, (ii) the ability to adapt dynamically to network topology changes, (iii) the guarantee for connectivity and low degree, and (iv) a realistic model for the RSSI.

Furthermore we have implemented and tested our algorithm on BTnodes, a platform for developing and prototyping wireless sensor network applications. In contrast to our work, most related topology control algorithms have not yet been implemented and evaluated on real sensor node platforms. In most cases, only simulation results are available that are often even based on impractical and unrealistic assumptions.

The strategy presented in this paper optimally leverages the properties of Bluetooth which we assume as an underlying communication medium for S-XTC. Bluetooth offers a connection oriented interface at the link-layer, i.e. the dominant cost of maintaining and powering a connected network originates in each connection. While the given case-study addresses some specific issues and constraints originating from the Bluetooth standard and devices used, the algorithm presented can also be applied on platforms with other radio technology or make use of another link-metric.

B. Application Scenario

The applications scenario discussed assumes a distributed wireless sensor network application with a single node acting as a central data sink and multiple distributed data sources. The application requires a connected network topology and should exhibit some robustness against failures and changes in the connectivity of the nodes. This is quite a typical scenario commonly found in many wireless sensor network applications. In more detail. Such a typical networking scenario where this algorithm fits in is discussed in the following.

A large amount of sensor nodes are deployed in a field. The nodes need to be deployed with sufficient density such that no disconnected clusters exist[1]. Besides that, no additional constraints on the deployment are made, which allows for both sparse and very dense regions.

In the scenario considered, the nodes remain stationary. However, it is possible that additional nodes join or occasionally leave accounting for cases of variable power supplies (e.g. batteries, solar-powered

[1]As studied in percolation theory [3], for randomly and uniformly distributed nodes, the network will be connected with very high probability if the network density is above 5 nodes per unit disk.

cell), failures or additional deployments at a later time. It is further possible that the quality of a connection between two nodes is temporally impaired by obstacles or other interference.

The nodes all have the capability to assess the strength of received signals at their radio. This is a reasonable assumption, as this feature has become a standard in almost all modern radio devices used on sensor nodes today.

When the nodes are turned on they start executing the S-XTC algorithm which then runs forever or until a certain state is reached. Such a state can be characterized by a condition being met at a node, e.g. availability of a route to a base-station, or by more specific performance metrics. Upon detection of the loss of such a state, the S-XTC algorithm can be re-enabled to further refine the network topology.

Topology control algorithms are typically implemented as a separate network layer in the embedded software of the nodes. The topology control layer notifies the higher layer about the links to the selected neighbors. The advantages of the S-XTC topology control for the application are as follows:

- *Connectivity:* The application can reach all other nodes through the selected neighbors.
- *Low Degree:* Even in very dense networks the number of selected neighbors is low, which allows for more efficient execution of communication operations such as routing or flooding. Furthermore a low degree is essential if protocols are used, that limit the maximal number of connected neighbors.
- *Energy Efficiency:* A node can save energy in two ways through topology control: For link oriented communication systems, such as Bluetooth, the power consumption generally grows linearly with the number of connections [6]. A lower degree then consequently leads to lower power consumption. For communication systems that can individually adjust the transmit power, energy is saved by reducing the transmit power to the level that is just needed to reach the worst selected neighbor and not the whole neighborhood anymore.

The original XTC algorithm is discussed in section III. The proposed algorithm is presented in section IV where S-XTC is also evaluated using simulation and the implementation on a testbed is discussed. A specific example of this application scenario using S-XTC is discussed in section V.

II. Related Work

Early work in topology control focused on the special case of randomly distributed nodes. Hou and Li [4] can be considered originators of topology control.

We do not consider centralized algorithms here, as they can not be implemented on distributed nodes, the typical scenario found in wireless sensor networks. In [9], distributed topology control algorithms are presented that assume that every node knows its location (e.g. from a GPS device). The CBTC algorithm [11] is based on directional information, which could also be achieved by directional antennas and beam-forming.

Topology control for Bluetooth ad-hoc networks is also referred to as scatternet formation. Previous research in this area has led to a number of algorithms, that can be distinguished according to the following three properties: (i) guarantee for a connected topology, (ii) guarantee for degree constraints, and (iii) the requirement that all devices are within each others range. In [1], the authors compared four scatternet formation algorithms and evaluated their performance. A more recent comparison with four additional algorithms can be found in [10]. According to the classification introduced earlier, only two of eight evaluated algorithms guarantee properties (i) and (ii) while not requiring all devices to be within each others range. One of them, the BlueMesh algorithm [7], forms a mesh topology. However, it only considers connectivity information for neighbor selection. In contrast to BlueMesh, the second algorithm, described in [10] additionally uses RSSI as link-metric and constructs a sparse mesh scatternet, also known as the relative neighborhood graph (RNG).

The XTC algorithm [12] uses an abstract link-metric, e.g. RSSI, to construct a graph corresponding to the RNG with the abstract link-metric as distance. Compared with previous solutions, XTC is probably the simplest, and therefore most practical topology control algorithm, that guarantees connectivity while not requiring all nodes to be within each others range. Additionally, XTC is a local algorithm, i.e. it does not require communication over multiple hops.

Algorithms such as the ones discussed above still have considerable drawbacks, that render them impractical for implementation. Firstly, most algorithms assume that all nodes start execution of their protocols simultaneously. Secondly, especially the RSSI-based algorithms assume that the estimation of the distance using RSSI has a fidelity of 1, i.e. $\text{RSSI}_1 > \text{RSSI}_2 \Rightarrow d_1 < d_2$. In practice, this is not the case as will be shown later on.

III. XTC Algorithm

A. Algorithm and Properties

Our algorithm is based on the recently published XTC algorithm [12], which is an extremely simple and strictly local algorithm. The authors of XTC claim that it is faster than any previous proposal and that it is currently the most realistic topology control algorithm available. Let us briefly describe here the basic functionality of XTC.

The algorithm consists of three main steps: Neighbor ordering, neighbor order exchange, and edge selection (see Alg. 1).

Algorithm 1 XTC

```
 1: Establish order $\prec_u$ over $u$'s neighbors in $G$
 2: Broadcast $\prec_u$ to each neighbor in $G$; receive orders from all
    neighbors
 3: Select topology control neighbors:
 4:   $N_u := \{\}$; $\widetilde{N}_u := \{\}$
 5:   while ($\prec_u$ contains unprocessed neighbors) do
 6:     $v :=$ least unprocessed neighbor in $\prec_u$
 7:     if ($\exists w \in N_u \cup \widetilde{N}_u : w \prec_v u$) then
 8:       $\widetilde{N}_u := \widetilde{N}_u \cup \{v\}$
 9:     else
10:       $N_u := N_u \cup \{v\}$
11:     end if
12:   end while
```

The algorithm operates on a initial graph $G = (V, E)$, which we refer here to as the *visibility graph*. For simulation and comparison, it is commonly accepted to assume that Graph G is a Unit Disk Graph (UDG), i.e. an Euclidean graph containing an edge (u, v) if and only if the normalized distance $|uv|$ is smaller or equal than 1. In other words: two nodes only "see" each other if they are within each others range.

In the first step, each network node computes a total order over all its neighbors with respect to decreasing link quality. This ordered link is then exchanged with all 1-hop neighbors. A node u selects the link to node v, if there exists no node w that has already been processed, that appears before u in the received order $\prec_v$ ($w \prec_v u$). After completion of the algorithm, the set N_u contains u's neighbors in the topology control graph G_{XTC}.

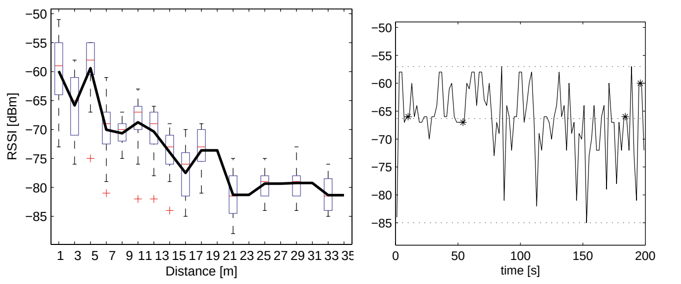


Fig. 1. (Left Fig.) RSSI over distance. The measurements are performed with two BTnodes at distance $|uv|$ and 1 m above the ground on a roof and alternated transmitting and receiving. The bold line is the average of 20 values. (Right Fig.) RSSI fluctuations over time of two stationary nodes. The stars denote unsuccessful RSSI estimates (nodes were not found by hci_inquiry) that are commonly found in measurements. Most algorithms account such sporadic outages poorly and thus are hard to transfer into realistic scenarios.

The main properties of the resulting topology of G_{XTC} are the following (see [12] for additional properties and proofs):

1) *Symmetry:* A node u includes a neighbor v in N_u if and only if v includes u in N_v.
2) *Connectivity:* Two nodes u and v are connected in G_{XTC} if and only if they are connected in the visibility graph G. Consequently, the graph G_{XTC} is connected if and only if G is connected.
3) *Bounded Degree:* Given an Euclidean Graph G, G_{XTC} has degree at most 6.
4) *Sparseness:* In an average-case simulation where nodes are placed randomly and uniformly on a Euclidean plane, the average degree of the nodes in G_{XTC} is constant at approximately 2.5.

B. Evaluation with a realistic RSSI model

XTC has been evaluated in [12] mainly on Euclidean graphs. When initially using RSSI values instead of the Euclidean distances for the ordering of neighbors in the first step of the algorithm, we obtained differing results. In order to understand this behavior we first measured the behavior of the $RSSI_u(v)$ on our target platform.

The results are shown in Fig. 1. We see that the RSSI values, even in average, are not a monotonously decreasing function of the distance, which can most likely be attributed to multipath reflections and other sources of interference. A second effect is that in a stationary scenario with constant distance between a transceiver pair, the values measured over time have a large variance. Other radio transceivers are known to exhibit similar behavior [8].

To investigate the consequence of these effects on the properties and the resulting topology we simulated the XTC algorithm. In contrast to the simulations presented in [12] we use the following RSSI model for the ordering the neighboring nodes instead of the Euclidean distance:

$$\begin{aligned} w \prec_u v &\Leftrightarrow RSSI_u(w) > RSSI_u(v) \\ RSSI_u(v) &= RSSI_1 - a \cdot \log(|uv|) - \sigma_{RSSI} \cdot x \\ a &= \frac{RSSI_1 - RSSI_{d_{\max}}}{\log(d_{\max})} \\ x &\sim \mathcal{N}(0,1) \end{aligned}$$

The reflections and the fluctuations of the RSSI are expressed in the model with the standard deviation σ_{RSSI} multiplied with a standard normal distributed random variable x. The model fits to our measurements with the values $RSSI_1 = -57\ dBm$, $RSSI_{d_{\max}} = -83\ dBm$, $d_{\max} = 40\ m$, and $\sigma_{RSSI} = 6\ dBm$. In the simulation 1000 nodes are placed randomly and uniformly in a square field. We altered σ_{RSSI} and the network density δ. The network density is defined as the average number of nodes within range ($d_{\max}$), which is equivalent to the number of nodes per unit disk. We simulated with densities from 1 to 30 nodes per unit disk and with a σ_{RSSI} from 0 to 13. The plots presented are averaged results of 1000 graphs.

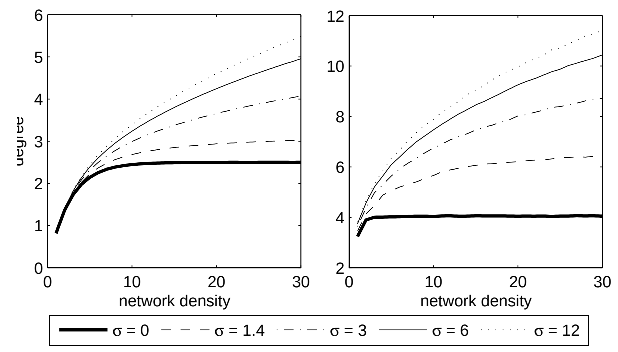


Fig. 2. Average (left) and maximal (right) degree vs. network density with varying RSSI standard deviation σ_{RSSI}.

Fig. 2 shows the average and maximal degree of the nodes in G_{XTC}. For $\sigma_{RSSI} = 0$ the neighbor ordering is the same as optained with the Euclidean distance in [12][2]. However for $\sigma_{RSSI} > 0$, the degrees are not bounded. They grow with the network density and with σ_{RSSI}. The number of edges also increases, as it is proportional to the average degree. For larger σ_{RSSI}, the link metric becomes uncorrelated to the distance. It was shown in [12] that for the case of arbitrary link weights (general weighted graphs) the degree of the nodes in G_{XTC} with n nodes can be $\Theta(n)$ and the number of edges $\Theta(n^2)$.

IV. S-XTC Algorithm

The XTC algorithm, as well as the majority of the other algorithms proposed, cannot be applied directly to our application scenario, because they contain rather impractical assumptions. In the case of XTC, we have identified the following problems.

1) Device Discovery is an unreliable operation. Especially in dense networks, it cannot be guaranteed that all neighbors are found, as required in the first step of XTC.
2) The nodes are not a priori synchronized and do not all start at the same time. However, an assumption in XTC is, that the tree construction phases are synchronized on all nodes.
3) It is assumed that the topology does not change.
4) For the XTC algorithm, the link weights are required to be symmetric. However, without additional communication we have $\text{RSSI}_u(v) \neq \text{RSSI}_v(u)$, and no guarantee for the connectivity can be given anymore.
5) A realistic RSSI is not a monotonous decreasing function of the distance ($\text{RSSI}_u(v) > \text{RSSI}_u(w) \Leftrightarrow |uv| < |uw|$). The maximal degree of a node is not bounded in that case.
6) As in XTC, every node needs to exchange its ordered list with all reachable neighbors, the time needed for this operation scales badly with the network density.

[2] $-a \cdot \log(|uv|) > -a \cdot \log(|uw|) \Leftrightarrow |uv| < |uw|$

Our proposed S-XTC algorithm addresses the above problems by providing 3 extensions.

- *Dynamic Adaptation:* With a general protocol change we solve problems 1-4.
- *Bounded Degree:* This extension can be used if a bounded degree is required (Problem 5).
- *Scalability:* With an additional heuristic, we reduce the number of required ordered list exchanges, while still guaranteeing the connectivity (Problem 6).

A. Wireless Communication

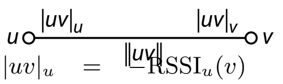
1) Communication Primitives: In order to foster an implementation of our algorithm not only for simulation but also on real sensor nodes, we give a more detailed description of the algorithm than such presented in related work. In particular, we describe what messages are exchanged and how they are handled. We use the following communication primitives:

discover: Upon request, beacon messages with the device ID are sent at the remote node. When such a message is received, the remote ID is stored and the RSSI value is measured. For Bluetooth, an integrated function (hci_inquiry) exists to perform this function.

connect/disconnect: This is only needed in a connection oriented medium such as Bluetooth, where a dedicated channel between two nodes must be established before messages can be sent.

send: We only consider point-to-point communication. The reason is that in dense networks the high interference would make a reliable broadcast transmission hard to achieve. In the case of Bluetooth, the communication is connection oriented.

2) Asymmetric RSSI link model: The algorithm uses a link-metric to decide whether a link is selected or not. However, if only local RSSI values are incorporated into this decision the resulting network topology might not be fully connected. The reason for these asymmetries is that the RSSI values measured by an adjacent node pair are usually not identical on both sides of the link. Therefore, we need to define a common link-metric. We use the following notation for a link in the visibility graph G



$$\begin{aligned} |uv|_u &= -\mathrm{RSSI}_u(v) \\ |uv|_v &= -\mathrm{RSSI}_v(u) \\ \|uv\| &= f(|uv|_u, |uv|_v) \end{aligned}$$

where $\mathrm{RSSI}_u(v)$ is the RSSI value that node v measures to node u. Increasing RSSI values denote increasing link quality. $\|uv\|$ is the combined link-metric derived from the exchange of the RSSI values at each node pair. It can be calculated using the function f on each side of a link as soon as the other value has been received. An evaluation using different functions for f will follow in section IV-D.

B. Dynamic Adaptation

1) Algorithm: The pseudo-code for this extension is given in Algorithm 2.

The main idea is that the decision, whether node u selects neighbor v or not, is taken directly when an ordered list $\prec_v$ is received at u. This results in the same selection as in the original XTC algorithm, because the decision only depends on $\prec_u$ and $\prec_v$ and not on other neighbors ordered lists. Furthermore, this allows for asynchronous operation at the nodes. Decisions can be taken independently from other neighbors state and directly on arrival of a neighbors ordered list. A further advantage is that the algorithm increases memory efficiency. In this way, there is no need to store ordered lists from all neighbors. In fact received lists can be discarded right after a single selection decision has been taken.

Algorithm 2 Adaptive XTC (for an individual node u)

```
process discovery
    repeat
        discover network changes
        determine order $\prec_u$
        add changed neighbors to ordered queue $Q$
        sleep($t_{\mathrm{sleep}}$)
    until termination event
end process
```

```
process connect
    loop
        if ($Q$ not empty) then
            $v$ = next node in $Q$
            connect(v) // if needed
            send($v$, orderlist($u$, $\prec_u$))
            remove $v$ from $Q$
        end if
    end loop
end process
```

```
msg_handler orderlist(node $v$, list $\prec_v$)
    remove $v$ from $Q$
    makeSymmetric($\prec_u$, $\prec_v$)
    if ($\exists w : w \prec_u v$ and $w \prec_v u$) then
        send($v$, nack($u$, $\prec_u$))
    else
        send($v$, ack($u$, $\prec_u$))
        inform higher layer that $v$ is selected
    end if
end msg_handler
```

```
msg_handler ack(node $v$, list $\prec_v$)
    makeSymmetric($\prec_u$, $\prec_v$)
    inform higher layer that $v$ is selected
end msg_handler
```

```
msg_handler nack(node $v$, list $\prec_v$)
    makeSymmetric($\prec_u$, $\prec_v$)
    inform higher layer that $v$ is deselected
    disconnect($v$)
end msg_handler
```

```
function makeSymmetric(list $\prec_u$, list $\prec_v$)
    determine $\|uv\|$
    $\prec_{u,\mathrm{old}} = \prec_u$
    update $\prec_u$ and $\prec_v$ using $\|uv\|$
    add all ($w : w \prec_{u,\mathrm{old}} v$ and $v \prec_u w$) to $Q$
end function
```

The algorithm has three processes that run in parallel. The *discovery* process is responsible for detecting network changes such as newly discovered nodes or changes in the link quality. The RSSI from all visible neighbors are repeatedly measured and stored with the corresponding node ID in a ordered list $\prec_u$. Based on the old and the new list, it can be determined which neighbors are affected by the change. Those neighbors are added to an ordered queue Q. It can not be guaranteed that every time all neighbors are correctly detected. The algorithm therefore continuously repeats these operations which ensures statistically, that all neighbors are found if the algorithm runs long enough.

The *connect* process continuously processes the entries in Q by sending $\prec_u$ to the neighbors. In contrast to the original XTC algorithm, the list $\prec_u$ not only contains the order of the nodes, but also the absolute link weight, which are needed later to calculate a combined symmetric link-metric.

The *message dispatch* process calls the appropriate handler for every incoming message. If an ordered list is received the function *makeSymmetric* is called. The combined link-metric $\|uv\|$ can be calculated from the local link-metric in $\prec_u$ and $\prec_v$. The nodes v and u are then reinserted according to $\|uv\|$ into $\prec_u$ and $\prec_v$ respectively. This reordering however, can influence the selection decision of the nodes w (line 5 in *makeSymmetric*) that are between v's old and new position in $\prec_u$. Finally, these nodes are then reinserted into the queue Q for processing.

2) RSSI Fluctuations: The algorithm reacts on network changes such as node addition and deletion and link losses. However, unnecessary changes due to RSSI fluctuations should be avoided as much as possible, since they cost energy and impair the network stability. In a field experiment with stationary nodes, deployed with 12 nodes per unit disc, we counted the number of connection establishments. Filtering the RSSI with a low-pass filter reduced the number of connection establishments by 20%. Additionally, we introduced a threshold value $\Delta RSSI_{th}$. The local neighbor order is only changed if the difference between the old and the new RSSI value is bigger than $\Delta RSSI_{th}$. Fig. IV-B.2 shows that an increased $\Delta RSSI_{th}$ stabilizes the network.

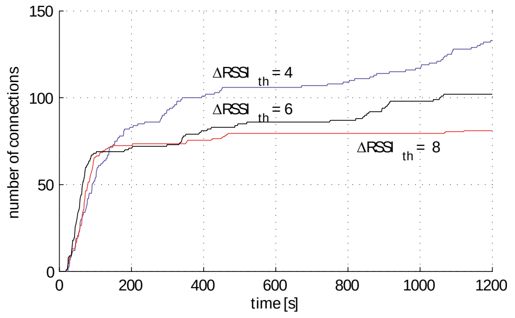


Fig. 3. Network establishment with 3 different $\Delta RSSI_{th}$ values in [dBm].

3) Evaluation on Testbed: We have implemented and evaluated S-XTC on a BTnode testbed with up to 40 nodes. Each node in the testbed locally stores the progress of the algorithm in a logfile, which is then retrieved by the base-station for evaluation. In order to measure the connectivity, we have added a process that periodically broadcasts a ping packet that is flooded on the network.

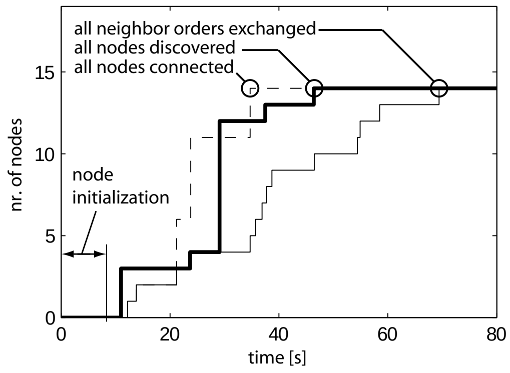


Fig. 4. S-XTC start-up measured on a single node running in a testbed with 15 nodes within transmission range.

Fig. 4 shows an example logging history of the S-XTC start-up on a node that has 14 neighbors. We have measured the time needed until the whole network is connected (dashed line), the time until all nodes are discovered (bold line), and the time needed to exchange the neighbor orders with all visible neighbors (solid line). In different test runs, we have deployed the nodes with different node densities. Average values, that have been found are listed in Table I.

node initialization time:
~ 9 s
time until all nodes are connected:
~ 33 s (independant of density)
time until all neighbor orders are exchanged:
~ 60 s (density 10)
~ 122 s (density 15)
~ 256 s (density 20)
~ 414 s (density 25)
time to add a new node to a existing network:
~ 11 s
time for a newly added node to exchange all neighbor orders:
~ 36 s (density 10)
~ 62 s (density 15)
~ 90 s (density 20)
~ 198 s (density 25)

TABLE I
TYPICAL DELAY MEASUREMENTS OF S-XTC.

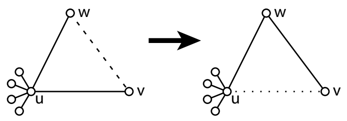


Fig. 5. When node u receives the ordered list $\prec_v$ it remembers node w as an alternative. If the degree of u is higher than the bound, the worst connection with an alternative is deleted. In this case u sends a message *swap(u, w)* to v to achieve the goal of global connectivity.

Considering that with Bluetooth, operations such as *inquiry* and *connect* itself require typically a few seconds to complete, and that the nodes are not synchronized, the S-XTC algorithm achieves quickly a connected topology. However, the device discovery and interferences are slowing down the exchange of neighbor orders, especially in dense networks.

C. Bounded Degree

The problem of the unbounded degree is not solved by the first extension proposed. The discovery process measures the signal strength to neighboring nodes periodically and averages the result. This has the effect that fluctuations are smoothed and the correlation between RSSI and distance is improved. Experiments have shown that we could reduce the standard deviation σ_{RSSI} by filtering from 6 to 3 dBm. However, it can be seen from Fig. 2, that for $\sigma_{RSSI} = 3$ the maximal degree being found using simulation is still above 8 for high densities. If nodes have limited number of possible connections such as the case on Bluetooth [5], a guarantee for the connectivity of the network topology derived cannot be given. We therefore propose a further extension using Alg. 3.

This algorithm replaces the message handler *orderlist* and *ack* of Algorithm 2 and adds a further handler *swap*.

We will now explain this algorithm using the example shown in Fig. 5. An ordered list $\prec_v$ is sent to node u, which starts to iterate through this list. Node u searches for common neighbors w, that appear before v in $\prec_u$. If no such node is found u selects v. Otherwise it is checked whether node w appears before u in $\prec_v$. If true, w is a better neighbor to both u and v, and the link uv is not selected. These two cases also appeared in the previous two algorithms. However the third case is new: if w appears after u in $\prec_v$, we have the situation shown on the left of Fig. 5. Node u can also reach v over w. This information is stored in an additional array *altNode*. Additionally,

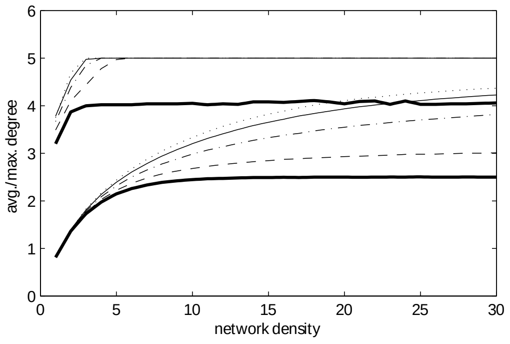


Fig. 6. Average (lower set of curves) and maximal (upper set of curves) degree of the Alg. 3 with a specified bound of 5.

node u remembers the connected neighbor ranked least using that has an alternative route *worstCandidate*. At the end of the message handler the current node degree is checked. If it is greater than the bound specified, it sends a *swap* message to *worstCandidate*. Node v handles the *swap* message, by adding node w to the queue Q and disconnecting the link to u.

The nodes of the disconnected link are not deleted from from the ordered list, because if found again during the discovery process, it would interpret them as new nodes. Instead, the nodes are only removed when sending an ordered list to an alternative node. Considering the example shown in Fig. 5 on the right side. When node u sends an ordered list to w (connect process, line 6), it sends $\prec_u \setminus v$; for all other nodes it sends $\prec_u$. So the edge uv is only deleted in the view of w. This information is stored in the array *deleted*.

Note that in the *swap* handler, the neighboring node w is not directly connected. Instead, it is added to the queue Q, in order to force node w to check if the new edge is really necessary.

The following theorem proves that this algorithm guarantees a maximal degree of 6, if G is a Unit Disk Graph.

Theorem 1 (Bounded Degree): Given a Unit Disk Graph, the topology control graph, obtained by Alg. 3 and a specified bound of 6, has at most degree 6.

Proof: Assume for contradiction that one node has degree bigger than 6. When the seventh neighbor was selected *worstCandidate* must have been *null* at the end of the *orderlist* or the *ack* message handler. This can only be the case if during the connection of all 7 neighbors no alternative node was found. Then this is only possible if there is no pair in the 7 neighbors that are also neighbors of each other. In a Unit Disk Graph, this would mean that no two adjacent edges enclose an angle less than $\pi/3$, which is only possible with at most 6 neighbors. ■

We have evaluated this extension with a specified bound of 5 in the same simulation setup as given for Fig. 2. The result is shown in Fig. 6, where we can see that none of the nodes violate the given bound. Compared to Fig. 2 the average degree is also improved, which is a result of the edges deleted.

In the simulation, where the Unit Disk Graph model is used, the degree bound can be set to 5. Note that in a real environment with interference, reflections, and obstacles, the UDG model is not realistic. In a worst case scenario, where a node has N neighbors, which do not see each other because of obstacles, the degree can be as large as N. However, this worst case too is an unrealistic assumption that could not be verified in test scenarios.

Algorithm 3 Bounded XTC (for an individual node u)

```
 1: process connect
 2:   loop
 3:     if ($Q$ not empty) then
 4:       $v$ = next node in $Q$
 5:       connect(v) // if needed
 6:       send($v$, orderlist($u$, $\prec_u \setminus$ deleted($v$))
 7:       remove $v$ from $Q$
 8:     end if
 9:   end loop
10: end process
```

```
 1: msg_handler orderlist(node $v$, list $\prec_v$)
 2:   remove $v$ from $Q$
 3:   makeSymmetric($\prec_u$, $\prec_v$)
 4:   repeat
 5:     $w$ = next node in list $\prec_v$
 6:   until (($w \prec_u v$) or endOfList)
 7:   if (endOfList) then // no common neighbor in $\prec_v$
 8:     degree = degree + 1
 9:     send($v$, ack($u$, $\prec_u$))
10:     inform higher layer that $v$ is selected
11:   else if ($w \prec_v u$) then // better common neighbor
12:     send($v$, nack($u$, $\prec_u$))
13:   else
14:     degree = degree + 1
15:     send($v$, ack($u$, $\prec_u$))
16:     inform higher layer that $v$ is selected
17:     altNode($v$) = $w$
18:     if (worstCandidate $\prec_u v$) then
19:       worstCandidate = v
20:     end if
21:   end if
22:   if ((degree > bound) and (worstCandidate not null)) then
23:     send(worstCandidate, swap($u$, altNode(worstCandidate)))
24:     deleted(altNode(worstCandidate)) = worstCandidate
25:     degree = degree - 1
26:   end if
27: end msg_handler
```

```
 1: msg_handler ack(node $v$, list $\prec_v$)
 2:   makeSymmetric($\prec_u$, $\prec_v$)
 3:   degree = degree + 1
 4:   inform higher layer that $v$ is selected
 5:   repeat
 6:     $w$ = next node in list $\prec_v$
 7:   until (($w \prec_u v$) or endOfList)
 8:   if ($w \prec_u v$) then
 9:   ... same as orderlist line 17-25
10: end msg_handler
```

```
 1: msg_handler swap(node $v$, node $w$)
 2:   deleted(w) = v
 3:   add $w$ to $Q$
 4:   inform higher layer that $v$ is deselected
 5:   degree = degree - 1;
 6:   disconnect($v$)
 7: end msg_handler
```

D. Improved Scalability in Dense Networks

The third extension addresses the scalability in dense networks. Suppose for example that a node has 100 neighbors. The XTC algorithm would require to exchange the ordered list with all of them, even if at the end maximally 5 nodes are selected. There will be a high interference, because all the 100 neighbors also want to exchange their order with each other. Here, broadcast communication is clearly inferior. However for link oriented communication, such as in Bluetooth, the situation is even worse. For every neighbor a connection has to be opened prior to the exchange and closed thereafter if it is not needed to support the network topology. The time for a connection setup in Bluetooth is in the order of a few

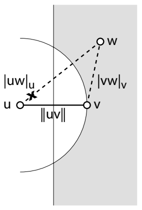


Fig. 7. Reduction of possible links for saving expensive exchanges: node u does not connect to node w if node w is in the shaded area.

seconds, if both nodes are ready. But if one node is already trying to connect to another node or performing a device discovery (inquiry), the delay can grow to tens of seconds, which is quite impractical. We have measured the time for the initial setup and found it to grow quadratically with the network density. This problem was also identified by the authors of [10].

1) Reduction Heuristic: With an additional heuristic, we can considerably reduce the number of exchanges required as shown in Fig. 7. Here, we use the notation of Sec. IV-A.2.

Suppose every node has a candidate list with all neighbor nodes, with which order lists have to be exchanged. This candidate list contains initially all neighbors. However, as soon as the first order lists have been exchanged, the candidate list can be reduced based on the received information. E.g. after node u has received the order list $\prec_v$ from v and calculated the common link weight $\|uv\|$, it can use additional information about a common neighbor w for the decision whether w is removed from the candidate list. In particular, node u removes node w from the candidate list, if

$$|uw|_u \quad > \quad |vw|_v \quad \text{, and} \tag{1}$$
$$|uw|_u \quad > \quad \|uv\|. \tag{2}$$

For Euclidean graphs, this is the case if node w is in the gray area in Fig. 7. An exchange of order lists between two nodes u and w is only prevented, if both u removes w and w removes u from their candidate lists.

We now show in the following theorem that with the proposed reduction applied to the XTC algorithm the connectivity is still guaranteed, if the common link weight is defined as $\|uv\| = \min(|uv|_u, |uv|_v)$.

Theorem 2 (Valid Reduction): The reduction of the ordered list exchanges applied to the XTC algorithm does not lead to a disconnected topology if the common link weight is defined as $\|uv\| = \min(|uv|_u, |uv|_v)$.

Proof: Since we know that G_{XTC} is connected, we prove that $G_{\text{XTC}} \subseteq G_{\text{reduction}}$. If an edge $uw \in G_{\text{XTC}}$ is not in $G_{\text{reduction}}$, it has not been selected due to a missing ordered list exchange. The only two possibilities for this are: (a) the edge uw is removed from the candidate lists of both u and w, and (b) the edge is not selected because another ordered list is missing.

Consider for contradiction an edge $uw \in G_{\text{XTC}}$ with

$$|uw|_u \leq |uw|_w \Rightarrow \|uw\| = |uw|_u. \tag{3}$$

In order to remove w from u's candidate list, there must have been an ordered list exchange between u and a third node v. Because $uw \in G_{\text{XTC}}$, the following equation must hold:

$$(\|uw\| < \|uv\|) \cup (\|uw\| < \|vw\|) \tag{4}$$

The conditions for the removal of w from u's candidate list are given in Eq. 1 and Eq. 2. Combining Eq. 1-3 we get:

$$\|uw\| \quad > \quad |vw|_v \tag{5}$$
$$\|uw\| \quad > \quad \|uv\| \tag{6}$$

With Eq. 4 and Eq. 6, we obtain:

$$\|uw\| < \|vw\| \tag{7}$$

Eq. 5 and 7 however, contradict the minimum definition:

$$\|uw\| < \|vw\| \leq |vw|_v < \|uw\| \tag{8}$$

This proves that (a) is not possible. For (b), consider the ordered list $\prec_u$ of a node u. A neighbor v can only be deselected due to a missing ordered list of a third node w, if w appears before v in $\prec_u$, but after v if $\|uw\|$ was known. This would mean that $|uw|_u < \|uw\|$ which is in contradiction to the definition. ■

2) Evaluation on Testbed: We have measured the startup times of S-XTC with the reduction heuristic and compared them to the previous results (see Fig. 8). We compare the time needed to exchange the neighbor order with all visible neighbors when all nodes have started synchronously (first two bars) and when only one node is added to a already existing network (last two bars).

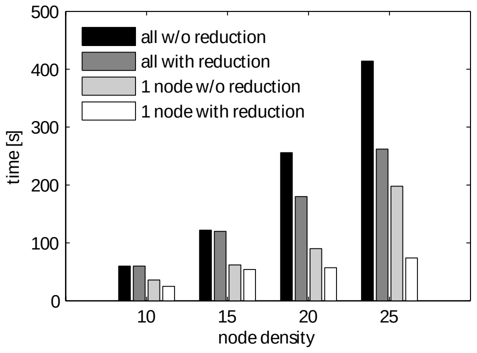


Fig. 8. Comparison of S-XTC startup time with and without the reduction heuristic.

For low densities, the startup times is dominated by the success of the device discovery and therefore the reduction heuristic has not much effect. With increasing node density the amount of necessary list exchanges among neighbors is reduced. We have observed that the number of effective exchanges with a node density of 25 is between 3 and 11, i.e. the reduction is on the order of 50%.

V. Case Study

Deployment-support networks (DSNs) have been proposed [2] as a non-permanent, wireless cable replacement for the development, testing and debugging of sensor network applications. As a general service, the DSN provides access to the sensor nodes with a wireless backbone network. One or more host computer are used to connect to the network and to communicate with the target sensor nodes using an underlying connected network topology.

The DSN topology is constructed and maintained using a simple tree-building algorithm based on a simple heuristic search and connect algorithm. The main problem with this solution is, that a link failure in the tree results in a disconnected network. Furthermore, because the algorithm is based on random search and connect, a number of low quality links are used in the backbone network resulting in sub-optimal topologies and significantly degraded reliability. With link failures happening at random, and in the worst case disconnecting

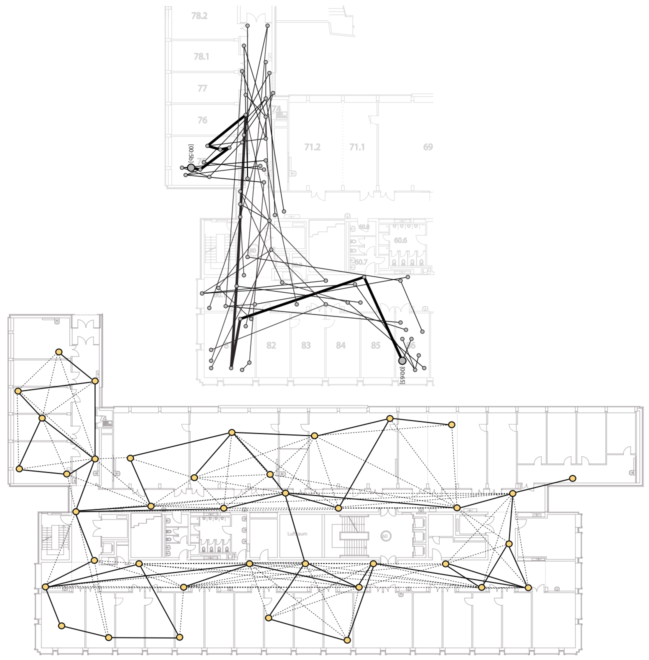


Fig. 9. Deployment of the DSN on an office floor using a previous tree based algorithm (left figure) and the new S-XTC algorithm using an RSSI link-metric (right figure).

the whole DSN from the host, the lack of redundancy can cause substantial disruption of service to the application.

We have successfully integrated S-XTC topology control in an implementation of a Deployment-Support Network using the BTnode platform. The application has been deployed on a testbed in an office floor comprising up to 50 nodes (see Fig. 9) and in various test scenarios (see section IV for details). Scenarios with differing amount and densities of nodes have been successfully tested. The network density in the setup shown in Fig. 9 varies between 2 and 12. The algorithm successfully connected the nodes into a connected network with an average degree of 2.9 and a maximal degree of 5.

The capabilities of the S-XTC network to adapt to gradual changes in the network and the environment as well as its improved resilience to link failures has improved the overall stability of the DSN application.

VI. Conclusion

In this paper we have analyzed existing approaches for network topology control and identified the key properties and eminent problems. Based on this analysis we have presented a practical topology control algorithm that combines:

- the guarantee for connectivity,
- energy efficiency through low degree topology control based on a single link metric,
- the ability to dynamically adapt to network and environmental changes, and
- a realistic model for asymmetric and inaccurate RSSIs.

The S-XTC algorithm is based on three extensions to the original XTC algoritm, that only render it into a reliable and practical topology control algorithm. The first extension *dynamic adaptation* addresses most shortcomings of the previous algorithm, i.e. operation in an unreliable environment, asynchronous startup and asymmetric link weights. The second extension *bounded degree* can guarantee an analytically proven bound on the node degree. The third extension *scalability* finally, significantly reduces the message overhead and computational complexity in dense networks.

The different extensions have been validated by means of analytic proofs and simulation. The simulation scenarios used have on the one hand been chosen similarly to the ones in the original XTC paper as to allow direct comparison and on the other hand to determine parameterizations and characteristics of the algorithm to facilitate implementation.

With the preparatory work from analysis, algorithm development and simulation at hand, and a preliminary test implementation of XTC, the S-XTC algorithm has been successfully implemented, on tiny, resource constrained wireless sensor network nodes.

The DSN application running on top S-XTC has proven the practicality of the algorithm in the case study presented in section V. Benchmarks and field tests have demonstrated it's performance: an increase in efficiency, lower message overhead, an improved selection of links as well as improved scalability over previous solutions. Even in high density scenarios (over 25 nodes within visibility) the time a first completion of S-XTC is reduced by approximately 1/3 over the algorithm without the proposed extensions. Likewise the amount of necessary neighbor list exchanges is reduced by about 50%.

Practical applications benefit from S-XTC due to advantages in the connectivity, low node degree and overall energy efficiency of the topology control algorithm.

References

[1] S. Basagni, R. Bruno, G. Mambrini, and C. Petrioli. Comparative performance evaluation of scatternet formation protocols for networks of Bluetooth devices. *Wireless Networks*, 10(2):197–213, Mar. 2004.

[2] J. Beutel, M. Dyer, L. Meier, and L. Thiele. Scalable topology control for deployment-support networks. In *Proc. 4th Int'l Conf. Information Processing in Sensor Networks (IPSN '05)*, pages 359–363. IEEE, Piscataway, NJ, Apr. 2005.

[3] O. Dousse, P. Thiran, and M. Hasler. Connectivity in ad-hoc and hybrid networks. In *Proc. of the 21th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2002.

[4] T. Hou and V. Li. Transmission range control in multihop packet radio networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[5] R. Kling, R. Adler, J. Huang, V. Hummel, and L. Nachman. Intel mote: Using Bluetooth in sensor networks. In *Proc. 2nd ACM Conf. Embedded Networked Sensor Systems (SenSys)*, page 318. ACM Press, New York, Nov. 2004.

[6] L. Negri, J. Beutel, and M. Dyer. The power consumption of bluetooth scatternets. In *IEEE Consumer Communications and Networking Conference*, pages 519–523. IEEE, Piscataway, NJ, 2006.

[7] C. Petrioli, S. Basagni, and I. Chlamtac. BlueMesh: degree-constrained multi-hop scatternet formation for bluetooth networks. *Mob. Netw. Appl.*, 9(1):33–47, 2004.

[8] J. Polastre, R. Szewczyk, and D. Culler. Telos: enabling ultra-low power wireless research. In *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, pages 364–369, 2005.

[9] V. Rodoplu and T. Meng. Minimum energy mobile wireless networks. *IEEE Journal on Selected Areas in Communications*, 17(8):1333–1344, Aug. 1999.

[10] E. Vergetis, R. Guerin, S. Sarkar, and J. Rank. Can bluetooth succeed as a large-scale ad hoc networking technology? *Selected Areas in Communications, IEEE Journal on*, 23(3):644–656, 2005.

[11] R. Wattenhofer, L. Li, P. Bahl, and Y.-M. Wang. Distributed topology control for power efficient operation in multihop wireless ad hoc networks. In *Proc. of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, 2001.

[12] R. Wattenhofer and A. Zollinger. XTC: a practical topology control algorithm for ad-hoc networks. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, pages 216–222, 2004.