REIHE INFORMATIK
TR-2005-006

TECA: A Topology and Energy Control Algorithm for Sensor Networks

Marcel Busse, Thomas Haenselmann,
and Wolfgang Effelsberg

Universität Mannheim
Praktische Informatik IV
A5, 6
D-68159 Mannheim, Germany

TECA: A Topology and Energy Control Algorithm for Sensor Networks

Marcel Busse, Thomas Haenselmann, and Wolfgang Effelsberg
Computer Science IV - University of Mannheim, Germany
University of Mannheim
Seminargebäude A5, D-68131 Mannheim, Germany
{busse, haenselmann, effelsberg}@informatik.uni-mannheim.de

***Abstract*— A main challenge in the field of sensor networks is energy efficiency to prolong the sensor's operational lifetime. Due to low-cost hardware, nodes' placement or hardware design, recharging might be impossible. Since most energy is spent for radio communication, many approaches exist that put sensor nodes into sleep mode with the communication radio turned off. In this paper, we propose a new Topology and Energy Control Algorithm called TECA. We will show the performance of TECA by means of extensive simulations compared to two other approaches. In terms of operational lifetime, packet delivery and network connectivity, TECA shows promising results. Unlike many other simulations, we use an appropriate link loss model that was verified in reality. By measuring packet delivery rates, TECA is able to adapt to different environments while still maintaining network connectivity.**

I. Introduction

Research in Sensor Networks has become more and more popular [1][8]. Advances in micro-sensor and radio technology enable low-cost hardware and embedded systems including wireless communication, sensing, and processing unit. At the same time, it allows densely populated sensor networks. Since in most cases sensor nodes will be battery powered, energy efficiency has to be taken into account [13]. Recharging might be impossible due to local conditions or might be inefficient because of low-cost hardware. Therefore, energy conserving algorithms are required that extend the lifetime of the entire network while still maintaining network operation.

The sensor node's energy consumption depends on many factors, e.g., usage of sensing components and communication technology. Some sensor nodes like the ESB [18] consume most energy for packet sending rather than receiving, whereas other nodes nearly spend the same amount of energy for both modes like the Mica motes [12]. All sensor nodes could save the most energy if they turned their wireless communication radio off and just stay in the sensing state. In order to maintain communication through the network, either some or all nodes have to wake up periodically or at certain times, or some nodes are prevented from turning their radio off at all. Of course, an important issue is the fact that the network will not be partitioned. Selecting which nodes can sleep and which nodes must be awake is the task of *Topology Control* or *Topology Management*.

In this paper, we present a new Topology and Energy Control Algorithm (TECA) that (i) extends network lifetime by putting nodes into sleep mode (radio turned off) and (ii) guarantees network connectivity. We will show by means of simulations that our algorithm saves more energy than other approaches and additionally minimizes packet loss by considering link qualities. Since in reality wireless links sometimes experience high packet loss rates, we take this issue into account. Unlike many other simulations, we will simulate packet loss on wireless links using an appropriate loss model that was verified in field studies.

The paper is organized as follows: In the next section, we briefly outline related work. Then we will describe our Topology and Energy Control Algorithm in Section III. Section IV gives a performance evaluation of TECA by exploring different parameter settings. A comparison with two other approaches is presented in Section V. Finally, Section VI ends with concluding remarks and an outlook on future work.

II. Related Work

Several energy-aware approaches have been proposed in the literature. Adaptive MAC layer protocols like S-MAC [25], T-MAC [21] and WiseMAC [10] allow nodes to turn off their radio if they do not participate in communication. All of these protocols have their major focus on reducing *idle listening* by introducing a duty cycle. In S-MAC, time is slotted in relatively large time frames consisting of an active and sleeping part. Only in the active part, data transmissions are possible. During the sleeping part, the radio will be turned off to save energy. The length of the active part is fixed whereas the sleeping time is under control of the application. As a consequence, nodes must listen to the channel during the whole active phase even if no transmissions are detected. T-MAC improves this situation by introducing a timeout value TA that determines the active part's length. If a node detects an activity on the channel after the timeout (data transmissions or collisions), it resets the timer. Otherwise, it assumes that the channel is idle and goes to sleep. Unlike S-MAC and T-MAC, WiseMAC is based on the preamble technique. In WiseMAC, active and sleeping phases are not synchronized among nodes. Each node informs its neighbors about its own wake-up time. With this knowledge, another node that has data to send just starts the transmission at the right time with a wake-up preamble.

In addition to energy-efficient MAC layer protocols, there exist several algorithms in the literature concerning topology control or topology management. Typically, two different approaches can be distinguished: Power control algorithms that minimize the node's transmission power and topology-based approaches that build a backbone of active nodes that participate in data delivery while all other nodes turn their radios off. Both approaches are orthogonal to the beforementioned energy-efficient MAC protocols and could be combined to further save energy.

The goals of power control approaches are to minimize interference, packet collisions and retransmissions as well as to improve spatial reuse. Ramanathan *et al.* [17] present two algorithms that maintain connectivity in the network by adjusting the maximal transmission power of a node. Both algorithms are centralized and based on global knowledge. Li *et al.* [15] propose LMST where each node builds a minimum spanning tree based on local information. The authors prove that their topology control algorithm preserves network connectivity with each node's degree bounded by 6. Furthermore, the constructed topology can be transformed into a bidirectional one by removing all uni-directional links without affecting connectivity. While the algorithm is limited to homogeneous networks, Li and Hou extend their

approach to heterogeneous networks where nodes may have different maximum transmission ranges in [14].

Another approach that relies on the number of adjacent neighbors is proposed by Blough *et al.* [3]. In k-Neigh, each node adaptively decreases its transmission power until just k symmetric links to adjacent nodes remain. Using distance estimations, the k nearest neighbors are determined that control the node's transmission power. The authors prove that their algorithm terminates after a total of $2n$ messages have been exchanged, with n nodes in the network. Also, they give an estimate for k that achieves connectivity with high probability. Interestingly, the value of k is only loosely depended on the number n of nodes in the network. Thus, knowledge about the exact value of n is not necessary. However, implementing k-Neigh in practice requires the ability of distance estimations that is not always possible.

Although many approaches claim to reduce interference by the spareness of the network as a result of power adjustment, Burkhart *et al.* [4] disprove this implication. They propose connectivity-preserving and spanner constructions that are interference-minimal. However, power control approaches may improve the network's capacity by reducing interference but concerning energy efficiency, their saving compared to idle listening will likely be marginal [19].

In contrast to power control, topology-based approaches exploit the fact that in densely deployed networks many nodes will be redundant. By turning the radio of these nodes off, a topology of active nodes is constructed. Because of omitting idle listening, those approaches will likely yield much energy conservation.

Xu *et al.* [24] have proposed Geographic Adaptive Fidelity (GAF), a topology control protocol that uses geographic positions of nodes to subdivide the sensor network into virtual grids. The grid size is chosen such that all nodes in one grid can communicate with all other nodes in adjacent grids. Given a transmission range r, the grid size a will be $r/\sqrt{5}$. Since nodes in a grid are considered equivalent from a routing perspective, just one node needs to be active with its radio turned on while all other nodes are sleeping. Thus, GAF exploits redundancy in the network and prolongs network lifetime with increasing node density. To balance out energy consumptions, sleeping nodes periodically wake up and rotate the role of the awake node among them. Since GAF assumes that each active node can communicate with active nodes in adjacent grids, the network will not be partitioned by the built backbone topology theoretically. There are not more partitions than in the underlying raw topology. However, this assumption does not always hold in reality where poor links with high packet loss occurs even if two nodes are close together. Furthermore, GAF relies on geographic information which are not always available.

Span [7] is another protocol that builds a topology backbone of forwarding "coordinators". It attempts to preserve the original network capacity and connectivity while reducing energy consumption. The node's decision of becoming a coordinator depends on remaining energy and the surrounding neighborhood. For example, a node becomes a coordinator if two unconnected adjacent coordinators can be connected. Like in GAF, nodes periodically wake up to balance energy consumption, and go to sleep if they do not have to join the forwarding backbone.

TMPO [2] shares many concepts with Span. However, its focus is on efficient communications rather than on energy conservations without providing sleeping nodes that turn their radio off. Based on a minimal dominating set (MDS), a backbone topology is constructed by transforming the MDS into a connected dominating set (CDS). TMPO uses the concept of clustering to build the MDS. By introducing *gateways* and *doorways*, these clusters will be connected in the CDS, guaranteeing network connectivity. A similar approach is presented in [22] with Cluster-based Energy Conservation (CEC).

Nikaein and Bonnet [16] emphasize the same motivation focusing on routing performance. They propose an algorithm that constructs a routing topology based on a forest. Each tree in the forest forms a zone that is maintained proactively. By considering the quality of connectivity, the set of non-overlapping zones are linked.

Dousse *et al.* [9] consider networks where nodes switch between on and off modes independently of each other. Since the on/off schedules are completely uncoordinated, the network might be disconnected most of the time. Assuming a store-and-forwarding routing mechanism, data is sent from nodes to a sink. Under some simplifying conditions, the maximum latency and variance is bounded. Moreover, the latency grows linear with the distance between a node and a sink depending on node density, connectivity range, and duration of active and sleeping periods.

Also putting nodes to sleep, ASCENT [5] builds a topology relying on the number of neighbors a node has discovered. However, only nodes whose packet loss is below a given threshold will be considered neighbors. The neighbor threshold NT determines if a node joins the backbone topology. If the node's number of neighbors is smaller than NT, the node will become active with its radio turned on. Otherwise, ASCENT assumes that there are enough active neighbors maintaining connectivity, and the node becomes passive. Although passive nodes do not join the backbone topology, they still overhear packet transmissions. In case the number of neighbors drops below NT or active nodes send help messages indicating poor link qualities to adjacent nodes, a passive node changes its state back to active. Since ASCENT also attempts to prolong network lifetime, each node has a passive timeout after it goes to sleep and turns its radio off. In addition, there is a sleeping timeout after sleeping nodes move back to passive again. Because of the simplicity of ASCENT and the fact that it mainly relies on the number of active neighbors, there might be situations where the network gets partitioned.

Like ASCENT, Naps [11] bounds the node degree in the constructed topology. While ASCENT tries to achieve a stable system, Naps puts nodes to sleep faster and more aggressively. The simulations show that most of the nodes are part of the largest connected component, but several distinct partitions are not unlikely.

Similar to Naps, AFECA [23] puts nodes into sleep mode, with the sleeping time being related to the number of neighbors. However, each node must have an accurate knowledge of the neighborhood size.

Schurgers *et al.* [20] propose STEM, a topology control protocol where nodes are equipped with a second radio channel for paging. The paging channel operates on a lower frequency and lower bandwidth conserving more energy. Nodes then shut down their main radio channel with their second radio turned on all the time. In case of packet delivery, the paging channel will be used to turn the main radio on again.

In this paper, we will present a topology-based approach without the need of a second communication radio and geographic information. Unlike most of the proposed algorithms, we perform simulations not based on a unit disk graph but on a realistic link loss model. We assume that all nodes are battery-powered and have the same transmission range. Moreover, we do not take mobility settings into account since most of the sensor network will likely be static. In the next sections, we will describe our proposed algorithm before we will compare it with GAF and ASCENT in Section V.

III. The Topology and Energy Control Algorithm

A. Basic Concept

Our proposed Topology and Energy Control Algorithm (TECA) is motivated by a clustering approach. Clustering the network means that each node is assigned to a *cluster* of nodes with one master node that acts as the *cluster head*. The cluster head is responsible for all

its assigned nodes and might perform special application tasks, handle data aggregation, control the medium access, or provide routing related functions. Once the network is divided into several clusters, TECA selects some nodes that act as *bridges* between two or more clusters. Thus, the entire network gets connected.

As shown in Figure 1, there are five states a sensor node can be in: *initial*, *sleeping*, *passive*, *bridge*, or *cluster head*. After a node is powered on, it is in the initialization state with its radio turned on until a timer T_i expires. In this state, nodes overhear packet transmissions, build a neighborhood table, and measure link qualities to adjacent nodes. After time T_i, a node changes its state to passive. Like in the initialization state, passive nodes overhear ongoing packet transmissions and keep their neighborhood table up-to-date. Additionally, in case of network disconnectivity, they will become *active*, either as a cluster head or as a bridge. Otherwise, they stay passive a time T_p until they go to sleep to save energy. We call a node sleeping if it turns its communication radio off. Other energy consuming components like sensing and processing units may still be turned on.

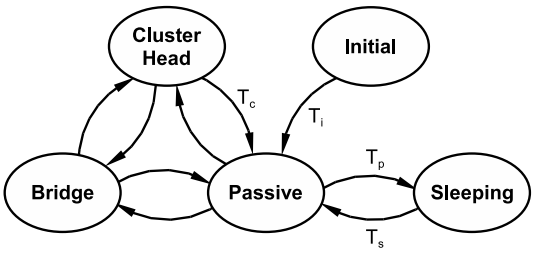


Fig. 1. TECA state transitions

Since TECA conserves energy by putting redundant nodes to sleep, a major challenge is to maintain connectivity in the network. Of course, a node with information about all nodes and links between them has the ability to build a well-connected topology. However, TECA should be *self-configuring* and should work in a distributed and localized fashion. Therefore, after clustering the entire network, TECA maintains connectivity by selecting nodes as bridges connecting different clusters. We will describe the cluster head and bridge selection process in detail in the next section. In addition to maintaining network connectivity, TECA attempts to select nodes joining the topology as sparsely as possible. Furthermore, it explicitly considers link qualities by measuring packet delivery rates. Especially in sensor networks, high packet losses are very common due to low-power radios, reflections, attenuation, and multipath/fading effects. Cerpa *et al.* [6] have shown that in an area of more than 50% of the communication range, there is no clear correlation between packet delivery and distance. In their measurements, link asymmetries occur with 5% to 30%. Further analyses have indicated that this is primarily caused by small differences in hardware calibration and energy levels between nodes.

B. TECA in Detail

The topology built by TECA is based on neighborhood information that nodes exchange periodically. These *beacons* are broadcast by non-sleeping nodes each time an announcement timer T_a expires. Except for sleeping nodes, all nodes send these beacons. Beacons contain the node's id, state, remaining energy, a timeout value, and 1-hop neighborhood information. However, only information about active neighbors, i.e., cluster heads and bridges, will be included into the packet. To identify asymmetric links and to provide other nodes with link qualities, loss information is added for each active neighbor. This information is based on local measurements of packet delivery and adapted by exponential smoothing over time. Since packet loss may be different depending on the transmission direction, both directions will be considered independently. Thus, we can identify asymmetric links through a difference in packet loss that is higher than a threshold LA.

While a node is in initialization state, it only sends out beacons every announcement time. After it changes its state to passive, it first sets a passive timer T_p after which it will turn its communication radio off and sleep to save energy. As long as a node is not sleeping, it checks its current state each time it receives or must send an announcement packet. The *CheckState(n)* function is shown in Algorithm 1.

Algorithm 1 $CheckState(Node\ n)$

```
1:  if (IsCluster(n)) then
2:     if (n.state ≠ clusterhead) then
3:        n.state = clusterhead
4:        set cluster head timer T_c
5:     end if
6:  else if (IsBridge(n)) then
7:     if (nstate ≠ bridge) then
8:        n.state = bridge
9:     end if
10: else
11:    if (n.state ≠ passive) then
12:       n.state = passive
13:       set passive timer T_p
14:    end if
15: end if
```

Once a node has lost its cluster head or determines that it is a cluster head itself, function *IsCluster(n)* will return true. Otherwise, the node verifies if it should be active to connect different clusters based on its 2-hop neighborhood information. If function *IsBridge(n)* also returns false, it remains or changes to passive mode. In the next two sections, we will look at both the cluster and bridge selection process in more detail.

1) Cluster Head Selection: The clustering selection process works as follows: As long as a node is not assigned to a cluster, it is a potential cluster candidate that it propagates to its neighborhood. Thus, the best *suitable* node is found, i.e., the node with the best cluster selection value, e.g., remaining energy. This node will then become a cluster head. All nodes in the cluster head's 1-hop neighborhood are assigned to it and are no longer cluster head candidates. With that mechanism, any node is either cluster head itself or assigned to one. Thus, after the cluster selection process, adjacent cluster heads are at most three hops apart, i.e., starting from an arbitrary cluster head, another cluster head can be reached in at most three hops.

Proof: Assume there are more than two nodes between two adjacent clusters. Consider the node that is 2 hops away from one cluster head c. Since this node is not a cluster head itself (otherwise the next cluster head would be just two hops away), it must be assigned to another cluster. Therefore, its cluster head is adjacent to c and thus three hops away. ■

Figure 2 depicts a possible cluster formation for a sample sensor network with 5 cluster heads. Each cluster is indicated by a circle around the cluster head that is equal to its radio transmission range. Although some nodes are in more than one cluster, they are assigned to just one cluster. Later, these nodes might become potential bridges (or bridge candidates) to connect two or more clusters with each other.

After a node is selected as a cluster head, it sets a cluster timer T_c during which it will not change its state. Due to load and energy balancing, it tries to find another cluster head it could join if T_c expires.

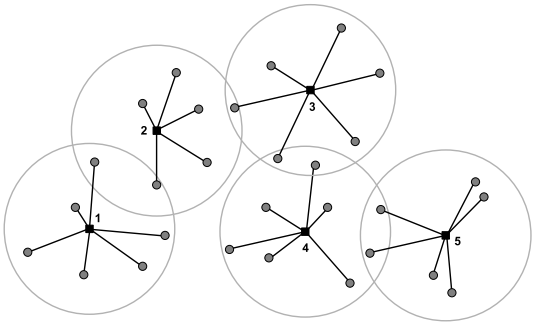


Fig. 2. Cluster formation

The running time of T_c is defined by

$$T_c = \min\{\alpha, energy_i\} \cdot E_i^{init} \quad (1)$$

with $\alpha \in [0 \dots 1]$ be the *cluster timeout factor*, E_i^{init} be the initially assigned energy in time units, and $energy_i$ be the fraction of node i's remaining energy.

The algorithm of the cluster head selection is shown in Algorithm 2. First, the best cluster head with respect to remaining energy is determined among all 1-hop neighbors. Note that we only consider neighbors with packet loss below a loss threshold defined by LT. In case of undecidability regarding a node's energy, we use the node's id as a breaking tie. If no existing cluster head is found, the node becomes a cluster head itself. However, if a cluster head is found whose cluster timeout timer was not expired and the node is a cluster head too, it just remains in its state if its remaining energy is higher (resp. has a lower id).

Algorithm 2 Cluster head selection: $IsCluster(Node\ n)$

if $(n.state = clusterhead \wedge time < n.T_c)$ **then**
  return true
end if
$c \leftarrow$ **null**
for all $(Neighbors\ m \in N : link(n,m).loss \leq LT)$ **do**
  if $(m.state = clusterhead \wedge m.T_c < time)$ **then**
    if $(!c \vee m.energy > c.energy \vee (m.energy = c.energy \wedge m.id < c.id))$ **then**
      $c \leftarrow m$
    end if
  end if
end for
if $(!c \vee (n.state = clusterhead \wedge (n.energy > c.energy \vee (n.energy = c.energy \wedge n.id < c.id))))$ **then**
  $c \leftarrow n$
end if
return $(c.id = n.id)$

The next step after clustering the entire network is to select bridge nodes to connect clusters with each other. Other nodes that are neither cluster heads nor bridges will later turn their radios off and sleep without participating in network communication. Then, the cluster head of the cluster sleeping nodes are assigned to is responsible for them. For example, cluster heads can store and later forward data packets to sleeping nodes, as soon as they wake up. Since sleeping nodes can still sense their environment, they must have the ability to communicate local events, e.g., to a data sink. In that case, they simply wake up and turn their radio on. As each node is assigned to a cluster head that maintains connectivity, even sleeping nodes can thus propagate local events to a data sink through the network.

2) Bridge Selection: After the cluster head selection process, all non-cluster heads remain passive until their passive timer T_p expires. During this phase, they listen to announcement packets of their neighbors. If a passive node is aware of the existence of more than one cluster, the node will become a bridge candidate.

Determining which of these nodes become bridges is a great challenge. There are three main requirements for selecting bridges: Bridges must connect different clusters in an optimal way, i.e.,

1) the packet loss between clusters should be minimized,
2) the connection should be long-lived, and
3) the number of selected bridges should be minimal to prolong the entire network's operation lifetime.

The basic idea of our bridge selection algorithm is to consider a node's 2-hop neighborhood as a graph with different link costs. Then, we compute the *Minimum Spanning Tree (MST)* but only take *virtual links* between cluster heads into account. Figure 3 shows an example of virtual links connecting all cluster heads. They are composed of one or more nodes that connect clusters best with respect to the above requirements.

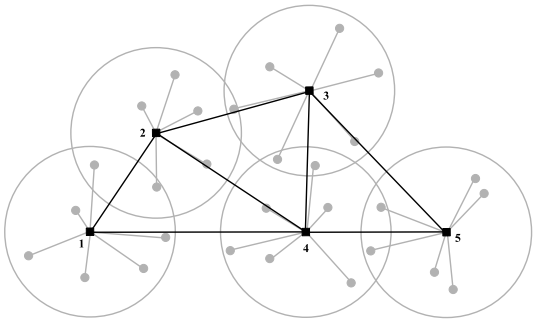


Fig. 3. Virtual cluster links

To reflect both packet loss and link lifetime, link costs are introduced. Later we will additionally use *penalty costs* in Section III-B.4 to minimize the number of active nodes.

The packet loss of virtual links will be computed as follows: Consider a virtual link containing k nodes $n_1 \dots n_k$ with n_1 and n_k be cluster heads. Let $loss_i$, $1 \leq i \leq k$, be the packet loss between n_1 and n_i and $loss_{i-1,i}$ the packet loss between two successive nodes. We then define

$$loss_i = \begin{cases} 0 & i = 1 \\ 1 - (1 - loss_{i-1}) \cdot (1 - loss_{i-1,i}) & i = 2 \dots k \end{cases} \quad (2)$$

Thus, the virtual link's loss is defined by $loss_k$. Also, the lifetime of a virtual link is defined by $lifetime_k$ with

$$lifetime_i = \begin{cases} energy_i & i = 1 \\ \min\{lifetime_{i-1}, energy_i\} & i = 2 \dots k. \end{cases} \quad (3)$$

where $energy_i$ is the fraction of the node's remaining energy.

The costs of a virtual link are defined by a priority function f that combines link lifetime and loss:

$$cost_i = 1 - f(lifetime_i, loss_i) \quad (4)$$

We will investigate this priority function in detail in Section III-B.3.

Figure 4 shows one possible mapping between virtual links and activated nodes - the bridges. Of course, there have been more nodes selected as bridges than necessary if we compute the MST on the entire network. However, all nodes have just a localized view of the network and thus are only able to take their 2-hop neighborhood information into account.

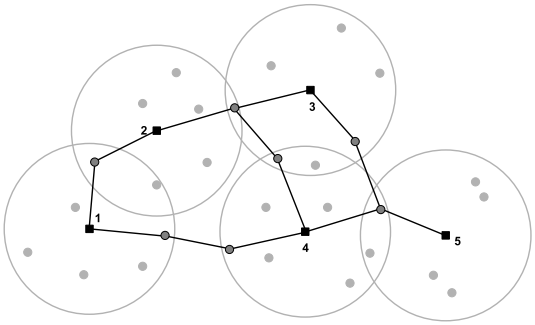


Fig. 4. Built Topology

The bridge selection algorithm is given in Algorithm 3 and 4. We will first have a look at the $IsBridge(n)$ function. If a node n acts as a cluster head with its cluster timer T_c still running, the node will not change its state. If the node just discovers less than two clusters in its 2-hop neighborhood N^2, it does not have to act as a bridge. Otherwise, we build a *Minimum Cluster Spanning Tree (MCST)* consisting of all selected cluster heads and bridges based on the MST algorithm using function $BuildTopology(n)$. Then, set $MCST^N$ contains all nodes that should be active to build the topology, i.e., cluster heads and bridge nodes. $IsBridge(n)$ returns true if $MCST^N$ contains the appropriate node n.

Algorithm 3 Bridge selection: $IsBridge(Node\ n)$

1: **if** $(n.state = clusterhead \vee$
$|\{Nodes\ m \in N^2 : m.state = clusterhead\}| < 2)$ **then**
2: **return false**
3: **end if**
4: $MCST^N \leftarrow BuildTopology(n)$
5: **return** $(n \in MCST^N)$

Algorithm 4 builds the MCST by running a priority search on the neighborhood graph. Nodes that are sleeping or dead, i.e., without remaining energy, are not considered. The crucial point of the algorithm is that the MST is not constructed with respect to all nodes but just to cluster heads, i.e., by just considering virtual links. We employ a heap implementing a priority queue that is used to get the next node (with minimal costs) starting at an arbitrary cluster head. Node's costs as a combination of lifetime and loss are computed according to Equation 2, 3, and 4. We will extend these costs by adding penalty costs in Section III-B.4.

The virtual links are built using the set $virtual\ link$ of node n. Each time we visit a new cluster head, all nodes along the virtual link are added to $MCST^N$. In addition, the node is marked as visited (with costs euqal to $-\infty$). The node's lifetime, loss, and virtual link set are reset.

After visiting node v, we consider node's v 1-hop neighborhood N^1. However, nodes that are already contained in v's virtual link set are skipped to avoid loops. For all other nodes, we compute the nodes' costs according to Equation 4.

In order to get a well-connected topology, we artificially increase the loss if the virtual link's loss is above the loss threshold LT. In such a case, we set the loss rate to $1 - \varepsilon$ with $0 < \varepsilon \ll 1$. Therewith, other virtual links might be preferred even if their costs (possibly influenced by lifetime) would actually be worse.

If an adjacent node w is reachable over node v with lower costs, w's fields $lifetime$, $loss$, $virtual\ link$, and $costs$ are updated. Additionally, the heap is updated, too. In case both costs are the same, we use the virtual link as a tie-breaker. Function $Cmp(link_1, link_2)$

Algorithm 4 $BuildTopology(Node\ n)$

1: **for all** $(Nodes\ m \in N^2 \cup \{n\})$ **do**
2: $m.costs \leftarrow (m.state \in \{sleeping, dead\})\ ?\ -\infty : \infty$
3: $m.virtual\ link \leftarrow \{m\}$
4: **end for**
5: $MCST^N \leftarrow \emptyset$
6: **for all** $(Nodes\ m \in N^2 : m.state = clusterhead)$ **do**
7: **if** $(m.cost \neq -\infty)$ **then**
8: $m.lifetime \leftarrow m.energy$
9: $m.loss \leftarrow 0$
10: $MCST^N \leftarrow MCST^N \cup \{m\}$
11: $prio\ queue.push(m, -\infty)$
12: **repeat**
13: $v \leftarrow prio\ queue.pop()$
14: **if** $(v.state = clusterhead \wedge v.costs \neq -\infty)$ **then**
15: $MCST^N \leftarrow MCST^N \cup v.virtual\ link$
16: $v.costs \leftarrow -\infty$
17: $v.lifetime \leftarrow v.energy$
18: $v.loss \leftarrow 0$
19: $v.virtual\ link \leftarrow \{v\}$
20: **end if**
21: **for all** $(Neighbors\ w \in N_v^1 : w \notin v.virtual\ link)$ **do**
22: **if** $(w.costs \neq -\infty)$ **then**
23: $lifetime \leftarrow \min\{v.lifetime, w.energy\}$
24: $loss \leftarrow 1 - (1 - v.loss) \cdot (1 - link(v, w).loss)$
25: $loss \leftarrow (loss > LT)\ ?\ 1 - \varepsilon : loss$
26: $costs \leftarrow 1 - f(lifetime, loss)$
27: **if** $(costs < w.costs \vee (costs = w.costs \wedge$
$Cmp(v.virtual\ link \cup \{w\}, w.virtual\ link) < 0))$
then
28: $w.lifetime \leftarrow lifetime$
29: $w.loss \leftarrow loss$
30: $w.virtual\ link \leftarrow v.virtual\ link \cup \{w\}$
31: $prio\ queue.update(w, costs)$
32: **end if**
33: **end if**
34: **end for**
35: **until** $(!prio\ queue.empty())$
36: **end if**
37: **end for**
38: **return** $MCST^N$

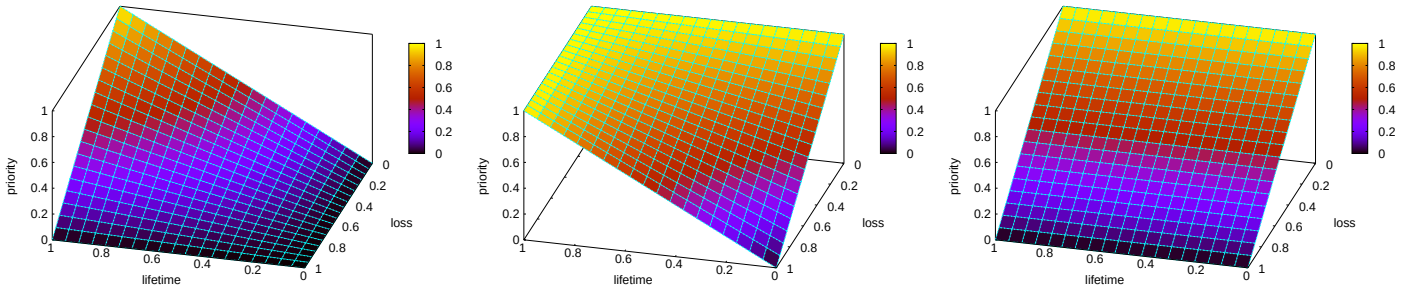
compares two virtual links returning -1 if the former is better, i.e., if

1) $|link_1| < |link_2|$, or
2) $|\{n \in link_1 : n.state = \{clusterhead, bridge\}\}| > \{m \in link_2 : m.state = \{clusterhead, bridge\}\}|$, or
3) $\sum_{n \in link_1} n.energy > \sum_{m \in link_2} m.energy$, or
4) $\min_{n \in link_1 \wedge n \notin link_2}\{n.id\} < \min_{m \notin link_1 \wedge m \in link_2}\{m.id\}$.

Note that a node's costs mainly depend on the priority function f. In the next section, we will have a deeper look at that function.

3) Lifetime Loss Model: The priority function f takes two parameters: remaining lifetime and packet loss. Based on both values, it shall determine the node's priority expressed by a value $\in [0..1]$. The higher the priority, the higher the probability to visit the appropriate node next. Certainly, there are many possible functions that would fulfill that requirement. We propose a two-dimensional linear function controlled by two parameters α and β defined by

$$\begin{aligned} f_{lifetime,0} &= (1 - \alpha) \cdot lifetime + \alpha \\ f_{lifetime,1} &= \beta \cdot lifetime \\ f(lifetime, loss) &= loss \cdot f_{lifetime,1} + (1 - loss) \cdot f_{lifetime,0}. \end{aligned} \tag{5}$$



(a) $\alpha = 0, \beta = 0$ (b) $\alpha = 1, \beta = 1$ (c) $\alpha = 1, \beta = 0$

Fig. 5. Priority function f with different α, β values

In Figure 5, the priority function is plotted for some α, β values. With parameters α and β, the weighting of lifetime and loss can be controlled differently. For example, a topology just based on remaining energy would be achieved by using $\alpha = 0$, $\beta = 1$. On the other hand, $\alpha = 1$, $\beta = 0$ would lead to a topology optimized for packet loss. That could also be achieved by using a linear combination $\gamma(1 - loss) + (1 - \gamma)lifetime$ resp. by using $\alpha = \gamma, \beta = 1 - \gamma$. But a linear combination would not satisfy the following requirements:

- Nodes with loss close to one should be considered worthless, and
- nodes with remaining energy close to zero should be considered worthless, too.

In this context, the term *worthless* means that the node's priority should be zero such that other nodes are preferred regarding the constructed topology. However, if we set $\alpha = 0$, $\beta = 0$, the priority function expresses that behavior as shown in Figure 5(a). In Section IV, we will investigate the influence of different α, β values on the performance of TECA by means of simulations.

4) Penalty Costs: As described in Section III-B.2, the third requirement of the bridge selection process is to minimize the number of selected bridges to save as much energy as possible. For example, consider the sub-graph depicted in Figure 6. If we take link loss (depicted as weights on each link) into account and neglect link lifetime, all nodes 6, 7, and 8 are selected as bridges. Then, MCST contains link $(2, 7, 3)$ with costs (packet loss) 0.0 and link $(2, 6, 8, 4)$ with costs 0.1. However, if we also took energy consumption into account and accepted higher link costs, link $(2, 7, 8, 4)$ could be an alternative since node 6 could sleep. Thus, the main question is how to tackle both cases.

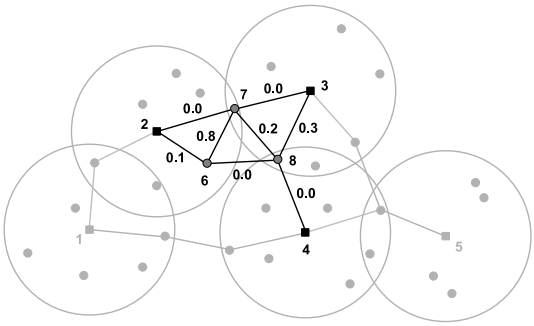


Fig. 6. Link costs

We handle this situation by introducing penalty costs. Nodes that are not yet in the topology will be penalized over nodes already selected. Again, consider Figure 6. Assuming the bridge selection algorithm shown in Algorithm 4 starts in node 2, link $(2, 7, 3)$ will be the first link added to MCST since first, all nodes 6, 7, 8 experience penalty costs. Now, node 7 becomes active (or remains active) and will be preferred over other nodes, i.e., it will no longer been penalized. Depending on the used penalty value PV, link $(2, 7, 8, 4)$ could get lower costs than link $(2, 6, 8, 4)$ in spite of higher packet loss.

We compute the penalty costs of a virtual link as follows: Consider a virtual link containing k nodes $n_1 \dots n_k$ with n_1 and n_k being cluster heads. Let $penalty_i$ be assigned to node n_i with $1 \leq i \leq k$. Let $0 < \varepsilon \ll 1$, and $0 \leq PV < 1$ be the penalty value. Then, we define

$$penalty_i = \begin{array}{ll} p_i & i = 1 \\ penalty_{i-1} + (1 - penalty_{i-1}) \cdot p_i & i = 2 \dots k \end{array} \tag{6}$$

with

$$p_i = \begin{cases} \varepsilon + PV & n_i \neq \{clusterhead, bridge\} \vee \\ & (n_i = \{clusterhead, bridge\} \wedge n_i \notin MCST^N). \\ \varepsilon & else \end{cases}$$

Thus, the costs function changes to

$$cost_i = c_i + (1 - c_i) \cdot penalty_i \quad i = 1 \dots k \tag{7}$$

with $c_i = 1 - f(lifetime_i, loss_i)$.

Constant ε is used to penalize longer links in terms of hops. Furthermore, passive nodes always receive penalty costs until they become active. For example, consider node 6 in Figure 6. If PV was high enough such that link $(2, 7, 8, 4)$ was added to MCST, node 6 would finally become a sleeping node. But if node 7 knew about another link connecting cluster heads 2 and 3 that was better than link $(2, 7, 3)$ and outside the scope of node 6, it might not become active at all if it selected link $(2, 6, 8, 4)$ to connect cluster heads 2 and 4. Then, node 6 as well as node 7 go to sleep, partitioning the network.

Therefore, just adding penalty costs to Algorithm 4 could lead to disconnected clusters depending on the order nodes are added to MCST, or more precisely on the order in which already activated nodes like cluster heads and bridges are added to MCST. Since the topology will be computed in a distributed and localized fashion, each node will likely build the MCST on different sub-graphs representing the node's 2-hop neighborhood. Thus, we cannot guarantee that for each node, Algorithm 4 starts with the same cluster head which could lead to different MCSTs.

For example, consider Figure 6 again. Let PV be 0.2 and the amount of remaining energy be negligible. Furthermore, assume that nodes 6, 7, 8 are bridges. The bridge selection algorithm of node 6 should start at cluster head 2 and that of node 7 at cluster head 4. First, let us consider the MCST of node 6. Since link $(2, 7, 3)$ has minimal costs,

it will be added to node 6's MCST first. Because we assumed that node 7 is already a bridge, it will no longer receive penalty costs. Therefore, link $(3, 7, 8, 4)$ is better than link $(2, 6, 8, 4)$ and added next. That terminates the algorithm with node 6 becoming passive.

The MCST of node 7 is built as follows: Since the bridge selection algorithm starts at node 4, link $(4, 8, 6, 2)$ is added first. However, the next best link is $(4, 8, 3)$. Node 7 assumes that it is redundant and becomes passive. Thus, both nodes 6 and 7 are passive likely leading to a partitioned network.

As we have seen, the graph's traversal order is crucial and could lead to different MCSTs if link/node costs are changed during the traversal. Therefore, we propose the following approach:

1) First, search for the best virtual link in the graph, i.e., the link with minimal costs.
2) Add that link to MCST.
3) Now search for the next best virtual link in the graph that is not yet contained in MCST.
4) If such a link exists, go back to step 2.

Referring to the last example, now both nodes 6 and 7 would add link $(2, 7, 3)$ to MCST first, independent from the traversal's starting point. Then, link $(2, 7, 8, 4)$ is added next since node 7 is already selected and does not receive penalty costs. Consequently, just node 6 becomes passive while node 7 remains in bridge state.

The enhanced bridge selection algorithm enabling penalty costs is presented in Algorithm 5 and 6. Set $MCST^L$ contains the selected MCST's links, whereas $MCST^N$ contains the selected nodes. Then, the MCST is built up successively. Virtual links are added to $MCST^L$ according to their link costs starting with the best link. The algorithm terminates if no further link is found, i.e., the MCST is complete. Let M be the number of cluster heads, then $MCST^L$ contains at most $M - 1$ virtual links. Thus, the priority search is only performed at most $M - 1$ times, too.

Algorithm 5 $BuildTopology(Node\ n)$

```
1:  MCST^N ← {Nodes m ∈ N^2 : n.state = clusterhead}
2:  MCST^L ← ∅
3:  repeat
4:    best link ← PrioritySearch(n, MCST^N, MCST^L)
5:    if (best link ≠ ∅) then
6:      MCST^N ← MCST^N ∪ best link
7:      MCST^L ← MCST^L ∪ {best link}
8:    end if
9:  until (best link = ∅)
10: return MCST^N
```

5) Sleeping Timeout: After selecting cluster heads and bridge nodes, all remaining nodes stay passive until their passive timer T_p expires. The intuition behind passive nodes is the ability to react to changes in the neighborhood as already activated nodes might become passive requiring other nodes to be active.

If T_p expires, a passive node goes to sleep with its radio turned off to save energy. However, the node must wake up at last at the cluster timeout to participate in rebuilding the topology.

Considering just the timeout of one's own cluster could lead to network partitions. For example, Figure 7 shows a topology of five nodes. Let node 1 and 3 be cluster heads, node 2 be a bridge, and node 4 and 5 be sleeping nodes. Furthermore, let node 4 be assigned to cluster 1 and node 2 and 5 be assigned to cluster 3. Assume the timeout of cluster 3 is before that of cluster 1. Then, node 5 wakes up first but will not find a cluster to join. Thus, it becomes a cluster head itself. If node 4 does not wake up at the same time, the network will get partitioned since node 5 does not have a connection to node 1 and 2.

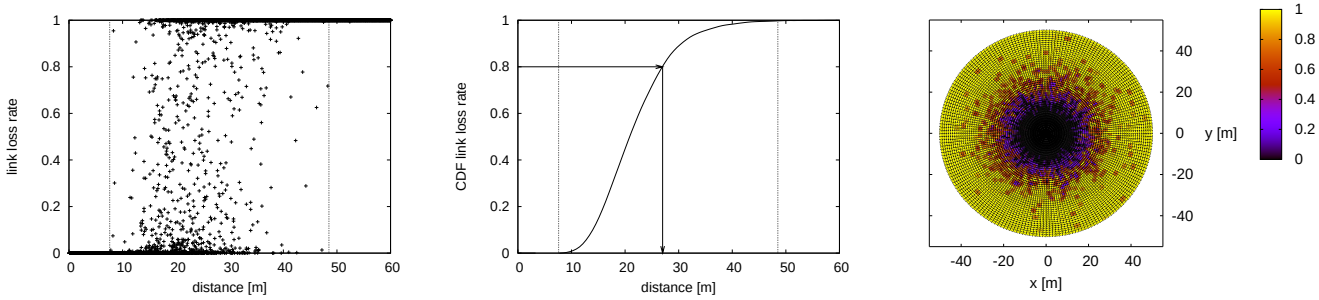
Algorithm 6 $PrioritySearch(Node\ n, MCST^N, MCST^L)$

```
1:  for all (Nodes m ∈ N^2 ∪ {n}) do
2:    m.costs ← (m.state ∈ {sleeping, dead}) ? −∞ : ∞
3:    m.virtual link ← {m}
4:  end for
5:  best link ← ∅
6:  best costs ← ∞
7:  for all (Nodes m ∈ N^2 : m.state = clusterhead) do
8:    if (m.cost ≠ −∞) then
9:      m.lifetime ← m.energy
10:     m.loss ← 0
11:     prio queue.push(m, −∞)
12:     repeat
13:       v ← prio queue.pop()
14:       if (v.state = clusterhead ∧ (v.costs ≠ −∞) then
15:         if (v.virtual link ∉ MCST^L ∧
            (v.costs < best costs ∨ (v.costs = best costs ∧
            Cmp(v.virtual link, best link) < 0))) then
16:           best link ← v.virtual link
17:           best costs ← v.costs
18:         end if
19:         v.costs ← −∞
20:         v.lifetime ← v.energy
21:         v.loss ← 0
22:         v.virtual link ← {v}
23:       end if
24:       for all (Neighbors w ∈ N_v^1 : w ∉ v.virtual link) do
25:         if (w.costs ≠ −∞) then
26:           lifetime ← min{v.lifetime, w.energy}
27:           loss ← 1 − (1 − v.loss) · (1 − link(v, w).loss)
28:           loss ← (loss > LT) ? 1 − ε : loss
29:           penalty ← ε
30:           if (w.state ∉ {clusterhead, bridge} ∨ (w.state ∈
              {clusterhead, bridge} ∧ w ∉ MCST^N)) then
31:             penalty ← penalty + PV
32:           end if
33:           penalty ← v.penalty + (1 − v.penalty) · penalty
34:           costs ← 1 − f(lifetime, loss)
35:           costs ← costs + (1 − costs) · penalty
36:           if (costs < w.costs ∨ (costs = w.costs ∧
              Cmp(v.virtual link∪{w}, w.virtual link) < 0))
              then
37:             w.lifetime ← lifetime
38:             w.loss ← loss
39:             w.penalty ← penalty
40:             w.virtual link ← v.virtual link ∪ {w}
41:             prio queue.update(w, costs)
42:           end if
43:         end if
44:       end for
45:     until (!prio queue.empty())
46:   end if
47: end for
48: return best link
```

On the other hand, if node 2 dies due to lack of energy before the timeouts of cluster 1 and 3, both clusters will get partitioned, too. In that case, node 4 must wake up to take the role of node 2.

Thus, a node has to wake up if

1) a cluster timeout of a known cluster in the nodes' neighborhood occurs, and



(a) Link loss rate (b) Cumulative density function of loss rate (c) Link loss map

Fig. 8. Link loss model

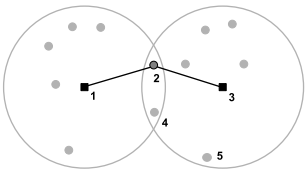


Fig. 7. Sleeping timeout example

2) a bridge runs out of energy with the network remaining connected if the considered node would be active.

Based on the MCST, the sleeping timeout is calculated as follows: First, the minimum of all known cluster timeouts is determined. Then, Algorithm 5 is used to identify *superior nodes* that suppress the passive node n from becoming active. For example, in Figure 7, node 2 is a superior node of node 4. However, Algorithm 5 needs to be modified regarding the set of nodes that are visited.

Let Ω be the set of active nodes (cluster heads and bridges) in the node n's 2-hop neighborhood. Then, the priority search is performed on $\Omega \cup \{n\} \backslash \{i\}$ for each bridge i. In case node n will be selected to be active, bridge i is a superior node of node n. Let Ω_C be the set of known cluster heads and Ω_S be the set of superior nodes of node n. Then, the sleeping timeout is calculated by

$$T_s = \min_{j \in \Omega_C \cup \Omega_S} \{timeout_j\} \tag{8}$$

with $timeout_j$ be the cluster timeout if $j \in \Omega_C$. If $j \in \Omega_S$, the timeout is determined by the amount of remaining energy.

IV. Performance Evaluation

As described in the last section, TECA's performance relies on user-defined parameters α, β, PV, and LT. In this section, we will investigate different parameter settings showing how they affect the topology built by TECA. Before we will present the simulation setup and results showing the impact of different parameter settings, we first have a look at the used link loss model.

A. Link Loss Model

The link loss model used in our simulations was published by Zuniga and Krishnamachari [26] who derived an analytical link layer model based on real data. They identify the existence of three distinct reception regions: *connected*, *transitional*, and *disconnected*. Figure 8(a) depicts these regions obtained from the analytical model. Within the connected region, actually no link loss occurs. Similarly, the disconnected region is characterized. However, within the transitional region that is quite significant in size, link losses are very common. Moreover, there is a high variance in loss rates, and asymmetric links exist. Especially in densely deployed sensor networks, more than 50% of links can be unreliable.

The *Link Loss Rate (LLR)* of distance d between a transmitter and receiver is defined by

$$LLR(d) = 1 - \quad 1 - \frac{1}{2} \exp^{-\frac{\gamma(d)}{2} \frac{1}{0.64}} \quad ^{\rho 8 f} \tag{9}$$

where γ is the *Signal-to-Noise Ratio (SNR)*, ρ the encoding ratio, and f the frame length. Equation 9 refers to a non-coherent frequency shift keying used as modulation technique with different encoding schemes. Several environmental and radio parameters are considered expressed by SNR, e.g., the environment's log-normal shadowing variance σ and the path-loss exponent η. In our simulations, we set $\sigma = 3.8$ and $\eta = 3.0$. The encoding ratio ρ is set to 1.0. For a deeper understanding of the model, please refer to [26].

The cumulative density function of the link loss rates is shown in Figure 8(b). Again, we can observe the three distinct regions with a maximum transmission range of about $50\ m$. However, many nodes within this range experience high link losses. Therefore, it is not advisable to use the maximum transmission range in Equation 11 for calculating the number of nodes N for a given density μ. Since node density is defined by the number of neighbors a node is connected to, r should rather be dependent on the link threshold LT. For $LT = 0.8$ that is used in our simulations, we get $r \approx 27\ m$.

Another view of link loss vs. distance is shown in Figure 8(c). With the transmitter at the center $(0, 0)$, the link loss rate to distant locations is colored. As depicted in Figure 8(a), nodes positioned within the transitional range receive high variance in link loss.

B. Simulation Setup

In our simulations, nodes are placed randomly on an area of size $100\ m \times 100\ m$ by using a uniform distribution function. All simulations are based on static networks as we expect most of sensor networks to be static. Let μ be the node density, i.e., the number of neighbors within a node's radio transmission range r. Then, the total number of nodes N placed on an area of size $A \times A$ can be approximated by

$$N = \mu \cdot \frac{A^2}{\pi r^2} \tag{10}$$

with a circular transmission range of size r.

However, Equation 10 does not take boundary effects of the area into account. Particulary for small sized areas, these effects are significant. Equation 11 represents a more precise calculation. A derivation is given in Appendix I.

$$N = \frac{\mu - 1}{p} + 1 \tag{11}$$

with $p \approx \frac{r^2(3.142A^2-2.667Ar+0.158r^2)}{A^4}$.

For each pair of nodes, we compute the link loss rate according to the presented link loss model. During packet delivery on the link layer, packets are randomly dropped with that loss rate. Packet collisions are not taken into account since they heavily depend on employed MAC schemes. Furthermore, we would like to concentrate on TECA's performance first.

All simulations are based on symmetric links with a neighbor threshold NT of 0.8. Modeling asymmetric links will be considered in future work.

In order to highlight the influence of different parameter values on TECA only, we assume that all nodes already know their neighbors and have sufficient link loss information. In addition, a node's remaining energy is uniformly distributed within $(0 \ldots 1]$ of the maximum energy value.

Then, for all parameter combinations, TECA is performed on each node until no further state changes occur. We carry out 20 simulation runs. For each run, the nodes' placements and generated link loss rates are the same.

C. Simulation Results

In order to investigate the performance of TECA based on different parameter settings, we are interested in the following metrics: *Mean Cluster Link Lifetime (MCLL)*, *Mean Cluster Link Loss Rate (MCLLR)*, and *Mean Number of Active Nodes (MNAN)*. MCLL and MCLLR are computed by taking all virtual links of the *global* MCST into account. Then, MCLL indicates the average lifetime until the network would get partitioned. Similarly, MCLLR defines the average quality cluster heads are linked by in the MCST.

The global MCST is built according to Algorithm 4 since penalty costs are out of concern. However, the algorithm is modified in two ways: (i) Only activated nodes, i.e., cluster heads and bridge nodes, will be considered, and (ii) neighbor tables of all nodes are taken into account to get a global MCST.

MCLL, MCLLR, and MNAN are defined as follows: Assume there are P partitions in the global topology of cluster heads and bridges. Let $MCST_p$ be the MCST of partition p. Let C_p be the number of cluster heads and B_p the number of selected bridge nodes in partition p. Therefore, the number of cluster links is equal to $C_p - 1$. Let n_{ijp} be the i-th node of cluster link j in partition p with $energy_{ijp}$ be the fraction of remaining energy and l_{jp} be the link length. Then, MCLL is defined by

$$MCLL = \frac{\sum_{p=1}^{P} \sum_{j=1}^{C_p-1} \min_{1 \le i \le l_{jp}} \{energy_{ijp}\}}{\sum_{p=1}^{P} (C_p - 1)}. \quad (12)$$

According to Equation 2, let $loss_{jp}$ be the loss of cluster link j containing l_{jp} nodes in partition p. MCLLR is then defined by

$$MCLLR = \frac{\sum_{p=1}^{P} \sum_{j=1}^{C_p-1} loss_{jp}}{\sum_{p=1}^{P} (C_p - 1)}. \quad (13)$$

For MNAN we get

$$MNAN = \frac{\sum_{p=1}^{P} (C_p + B_p)}{N} \quad (14)$$

with N equal to the total number of nodes.

Varying parameters α, β, and PV, we run TECA on each node until a stable topology is reached. All simulations are carried out using a node density of 10 and a loss threshold LT of 0.8. The setup is the same as described in Section IV-B.

Figure 9 depicts the simulation results for $PV = 0$. Except for $\alpha = 1$, the amount of MCLL is quite the same. As expected, the highest value is achieved if we just take lifetime into account and neglect link loss expressed by $\alpha = 0$, $\beta = 1$. However, as long as lifetime is considered at all, i.e., $\alpha < 1$, nodes with low link loss can be prioritized with respect to remaining energy without using node's id as a tie-breaker. Particular in densely deployed networks, there are likely many nodes receiving low or no link loss which could achieve better link lifetimes. Therefore, in case of $\alpha = 1$ and low loss rates, many decisions are just based on node ids leading to poor cluster link lifetimes.

The cluster link loss without penalty costs is shown in Figure 9(b). Actually, for most values of α, β, we get a very low MCLLR with $\alpha = 1$, $\beta = 0$ performing best. If just lifetime is taken into account ($\alpha = 0$, $\beta = 1$), the resulting MCLLR is quite bad. Direct links between cluster heads usually have the best link lifetime since cluster heads are selected regarding remaining energy. However, loss on these links is very high, i.e., higher than LT.

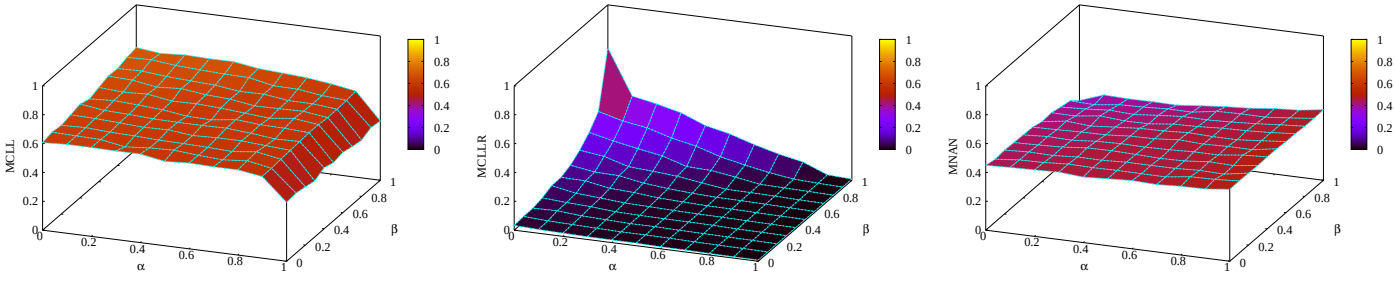
Note that MCLLR for $\alpha = 1$, $\beta = 1$ is quite the same as for $\alpha = 0$, $\beta = 0$. Since the network is dense enough that many nodes receive low link loss and the amount of energy varies among nodes considerably, nodes with low loss will likely get higher priorities than nodes with high link lifetimes (see Figure 5(b)). Using the definition in Equation 2 and 3, that means that $loss_i < 1 - lifetime_i$ applies to many nodes n_i.

The amount of active nodes, i.e., cluster heads and bridge nodes, is shown in Figure 9(c). Except for $\alpha = 1$ and $\alpha = 0$, $\beta = 1$, MNAN is about 0.4%. In case of $\alpha = 1$, many nodes are selected regarding their id due to the fact that they likely receive low link loss (because of the dense deployment). As soon as α is below 1, loss is taken into account leading to reusing nodes with low loss values instead of arbitrarily selecting nodes.

For $\alpha = 0$, $\beta = 1$, MNAN is lower than for all other combinations. In that case, just link lifetime is taken into account that is best if direct links between cluster heads are selected even if they experience high loss rates. Using direct links leads to fewer active nodes since bridges are not required.

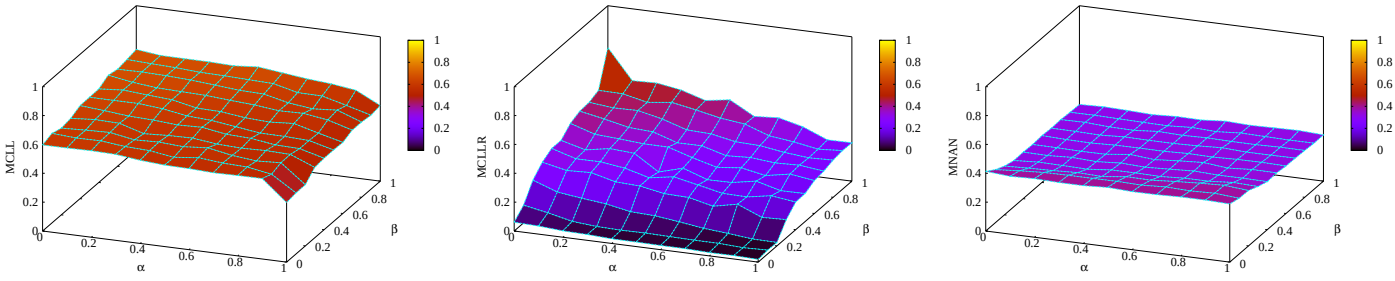
Reducing the number of selected bridges is only possible up to a certain amount since TECA always attempts to maintain connectivity. Therefore, a minimum number of bridge nodes are needed. Figure 10 shows the same graphs for penalty costs 0.8. As expected, MNAN decreases with increasing PV. However, the influence on MCLL is low. On the other hand, MCLLR changes significantly with increasing penalty costs. Only for $\beta < 0.2$, MCLLR is roughly as optimal as in Figure 9(b). Consequently, more direct cluster links are selected because selecting bridges causes penalty costs preferring direct links even in case of higher loss rates. That is also shown in Figure 10(c). However, if link loss is taken into account sufficiently ($\beta < 0.2$), penalty costs are compensated by the fact that direct cluster links characterized by high lifetime and loss get worse priorities. Thus, more nodes become bridges achieving better link loss rates.

As the simulation results have shown, the parameter settings are highly application dependent. However, low link loss will likely be the main issue for most applications. Therefore, it should be considered with high priority, i.e., β should be set to 0. The amount of lifetime influenced by α is a trade-off. We suggest to set $\alpha = 0$ that works best in densely as well as sparsely deployed networks. Especially at the end of the entire network lifetime, the network stability is much better if nodes with low remaining energy are downgraded such that nodes with more remaining energy are preferred (even in case of higher link losses). Moreover, the network's energy consumption will be balanced among all nodes better.



(a) MCLL (b) MCLLR (c) MNAN

Fig. 9. TECA's performance for $PV = 0$



(a) MCLL (b) MCLLR (c) MNAN

Fig. 10. TECA's performance for $PV = 0.8$

V. Comparison with other Approaches

We have conducted extensive simulations comparing TECA to other proposed approaches, namely GAF and ASCENT. While GAF uses node position information to build up the topology, ASCENT is just based on a neighbor threshold NT (see Section II). In contrast to TECA, neither one of them take network partitions into account explicitly. Preventing network partitioning is tackled by a smaller grid size in GAF and a higher neighbor threshold in ASCENT. A first expression on how the built topologies of active nodes might look like is given in Figure 11.

All algorithms are simulated using the same set-up as in the last section, varying node density, initial energy, and cluster timeout factor. Each simulation point in the graphs represents the average of 20 runs. The simulation area is 100 m × 100 m. According to the link loss model presented in Section IV-A, the radio range for getting a 80%-connectivity is about 27 m. Table I gives an overview of all parameter settings.

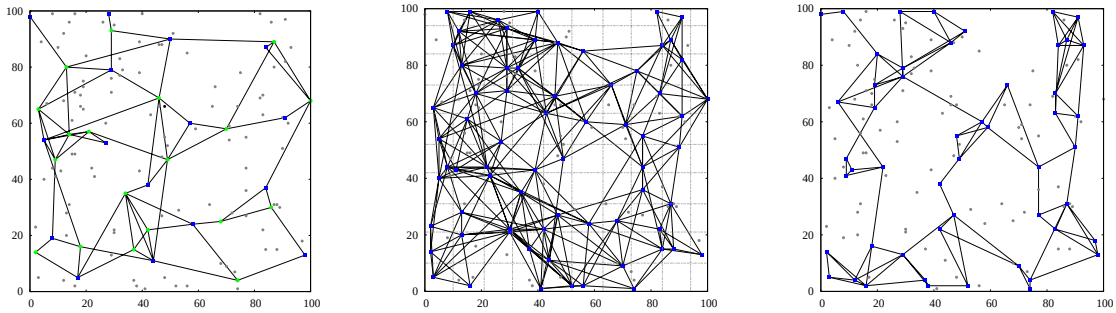
By means of simulations, we are interested in the following issues:

1) How many nodes are selected as cluster heads, bridges, passive and sleeping nodes? How does the topology change over time? How many nodes are selected by GAF and ASCENT?
2) How is the energy consumption over time? How much can the network's operational lifetime be extended?
3) Are there still situations where network partitions occur? How loss-resistent is the topology regarding end-to-end packet delivery?
4) How does the network lifetime scale with respect to node density, initial energy, and cluster timeout factor?

To get first answers to these questions, we focus our simulations on the node's selection process, mainly responsible for energy consumption. Furthermore, we use a very simple energy model that just differentiates energy consumption of nodes with their communication radio on vs. nodes with their radio turned off. In addition, MAC layer behavior has not been taken into account, e.g., packet collisions do not occur in the simulation. One reason for that is computational complexity. On the other hand, we are interested in results on a high level at first. We will consider MAC-related issues in future work.

The first simulation set considers the node's selection process and its evolution over time. All results are presented for TECA, GAF, and ASCENT separately. TECA is run with $\alpha = 0$, $\beta = 0$, and $PV = 0.8$ representing an appropriate trade-off between link loss and lifetime. Once a second, active and passive nodes send an announcement packet that contains its id, state, energy, timeout, and 1-hop neighborhood of active nodes together with link loss information. To keep neighbor information consistent, a sequence number is included, too. Nodes are considered dead if no announcement is received within $T_d = 10$ s. For reducing computational complexity, nodes do not verify their state each time a control packet is received but set a timer first. The verification timeout T_v is set to 200 ms. If it expires, the node's state is verified bundling all neighborhood changes that occurred within T_v. The passive timeout T_p after a node goes from passive to sleep is 2 s. However, the sleeping timeout T_s depends on the cluster timeout factor and the timeouts of adjacent nodes. It is calculated by each node individually according to Equation 8. To get a first link loss estimate, an initialization phase is carried out that lasts $T_i = 10$ s.

According to $r/\sqrt{5}$, GAF uses a grid size of about 12 m getting a connectivity of about 80%. Like TECA, it employs the same timeouts T_i, T_a, T_p, T_v, and T_d. However, the sleeping timeout T_s is calculated



(a) TECA (b) GAF (c) ASCENT

Fig. 11. Built topology by TECA, GAF, and ASCENT

differently. Each grid can be considered as a cluster with the activated node be the cluster head. This node computes its cluster timeout as in TECA. Since there are no bridge nodes, T_s is just set to T_c. Furthermore, a node can go to sleep as soon as it encounters a node with more energy. Announcement packets do not include the 1-hop neighborhood but additionally contain the node's position.

On the other hand, ASCENT uses the same announcement packets like TECA. The sleep timeouts are calculated by considering the timeouts of all active neighbors. Each active node sets its timeout according to Equation 1 (like cluster heads in TECA). Then, T_s is the minimum of all active neighbor's timeouts. The neighbor threshold NT controlling ASCENT's topology is set to 5 as recommended by the authors. All other parameters are the same as in TECA.

A. Investigating TECA, GAF, and ASCENT over Time

In the first set of simulations, we employ a node's initial energy value of 100 s, i.e., after 100 s a node will die if it keeps its radio turned on all the time. Furthermore, all algorithms employ a cluster timeout factor of 0.5, balancing network lifetime and energy deviation among all nodes.

Figure 12 depicts the percentage of nodes in active, passive, sleeping, and dead state over time for TECA, GAF, and ASCENT. Active nodes are nodes that build the topology, i.e., cluster heads as well as bridges in TECA. Passive nodes still have their radio on and probe the network of becoming active if necessary. Sleeping nodes turn their radio off and change back to passive after their sleeping timeout T_s. Dead nodes are nodes that ran out of energy.

Parameter	Setting
Simulation area	$100\ m \times 100\ m$
Simulation runs	20
80% radio range	$27\ m$
Loss threshold LT	0.8
Number of retransmissions	2
T_{init} (T_i)	$10\ s$
$T_{announcement}$ (T_a)	$1\ s$
$T_{passive}$ (T_p)	$2\ s$
T_{verify} (T_v)	$200\ ms$
T_{dead} (T_d)	$10\ s$
Node density	$10 \ldots 50$
Initial energy E^{init}	$10^2\ s \ldots 10^5\ s$
Cluster timeout factor	$0.1 \ldots 1.0$

TABLE I
SIMULATION SETTINGS

Compared to GAF, TECA and ASCENT select fewer active nodes leading to more nodes that sleep and save energy. Due to an initial energy value of 100 s and a cluster timeout factor of 0.5, each 50 s sleeping nodes wake up. Since all nodes are powered on at the same time, these cycles are more or less synchronized. If MAC collisions were taken into account, wake-up times would be spread over time to avoid these synchronization effects.

In TECA, most of the cluster heads change their state to balance energy consumptions among all nodes after cluster timeouts. Each time the topology changes, it takes some time until the topology is stable. This time depends on the announcement time T_a, the time nodes stay passive, and on how fast nodes verify their state after neighborhood changes. However, the amount of active nodes remains constant until too many nodes die.

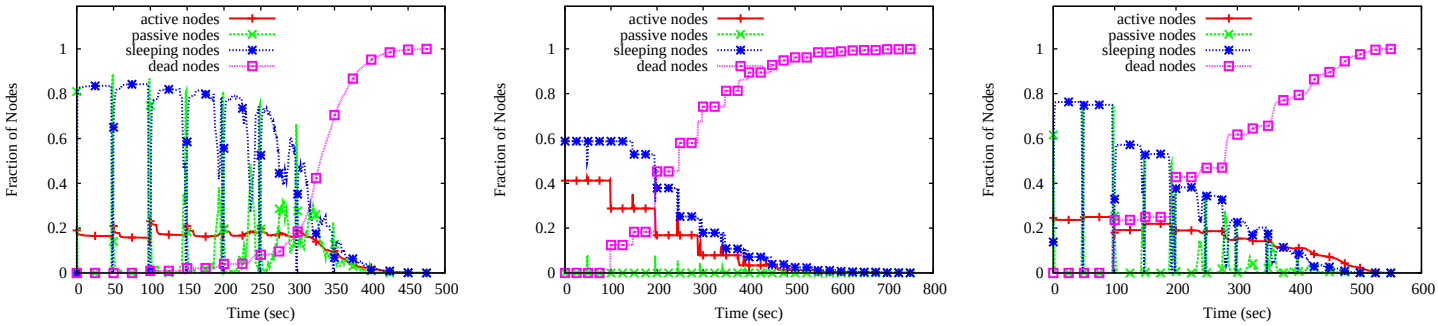
Since GAF is just based on the number of selected nodes in grids, the number of active nodes decreases over time with the dying of entire grids. However, ASCENT selects the same amount of active nodes that remains constant over time because of a fixed neighbor threshold.

Observing the evolution of dead nodes, we see that nodes in TECA stay alive for a longer time than in GAF and ASCENT. However, getting closer to the end, most of the alive nodes are activated preventing network partitioning. This leads to a shorter remaining network lifetime.

In contrast to that, GAF benefits from densely deployed grids since only one node will be active while all other nodes sleep. The grid with the maximum number of nodes determines the time until all nodes in the network are dead. Therefore, GAF archives the longest simulation time even if most of the nodes are already dead.

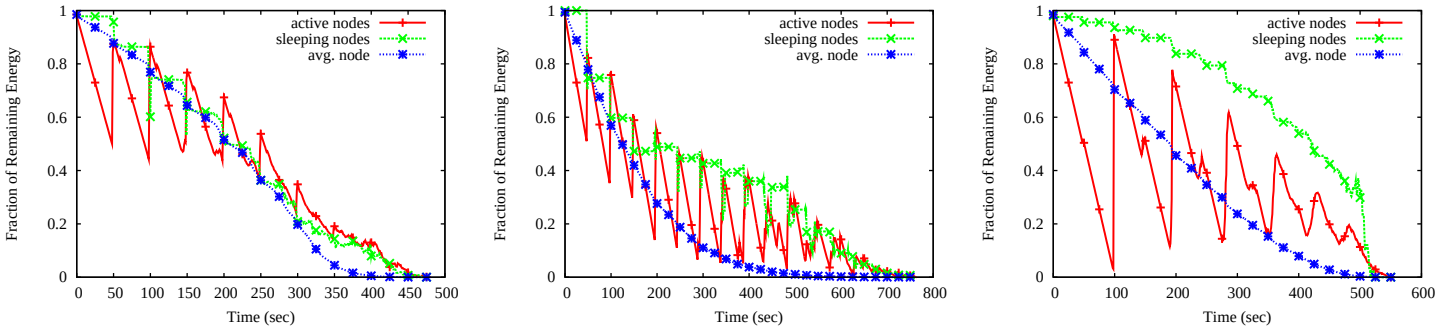
The remaining amount of energy over time is shown in Figure 13. While sleeping nodes do not consume energy, the energy of active nodes constantly decreases until the topology is rebuilt. Since in ASCENT active nodes stay active until they run out of energy, the node's energy deviation is worse than for TECA and GAF. Particulary at the end of the network lifetime, TECA performs best regarding balancing energy consumption among all nodes. At that time, it is likely that many link lifetimes are shorter than the remaining cluster's energy requiring sleeping nodes to wake up earlier to maintain connectivity.

Figure 14 depicts information about a node's neighborhood. The number of neighbors refers to nodes that are not dead and within the 1-hop neighborhood, averaged over all nodes. The *number of active neighbors* and the *node degree* refer to active neighbors only. While the number of neighbors is averaged over all nodes, the node degree considers just active nodes. Regarding the number of neighbors, TECA performs best. Since it balances energy consumption among nodes better than GAF and ASCENT, the number of neighbors remains



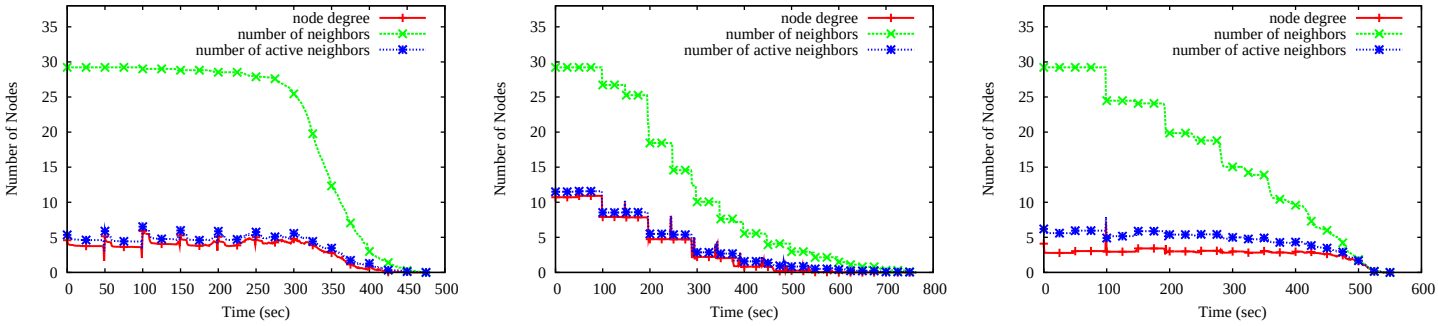
(a) TECA (b) GAF (c) ASCENT

Fig. 12. Fraction of different node types vs. time



(a) TECA (b) GAF (c) ASCENT

Fig. 13. Fraction of remaining energy for different node types vs. time



(a) TECA (b) GAF (c) ASCENT

Fig. 14. Node degree and number of neighbors vs. time

constant a longer time.

The number of active neighbors is quite the same in TECA and ASCENT. It is constant over time, in contrast to GAF where it depends on the lifetime of single grids. However, the node degree in TECA is higher than in ASCENT, i.e., the topology is better connected.

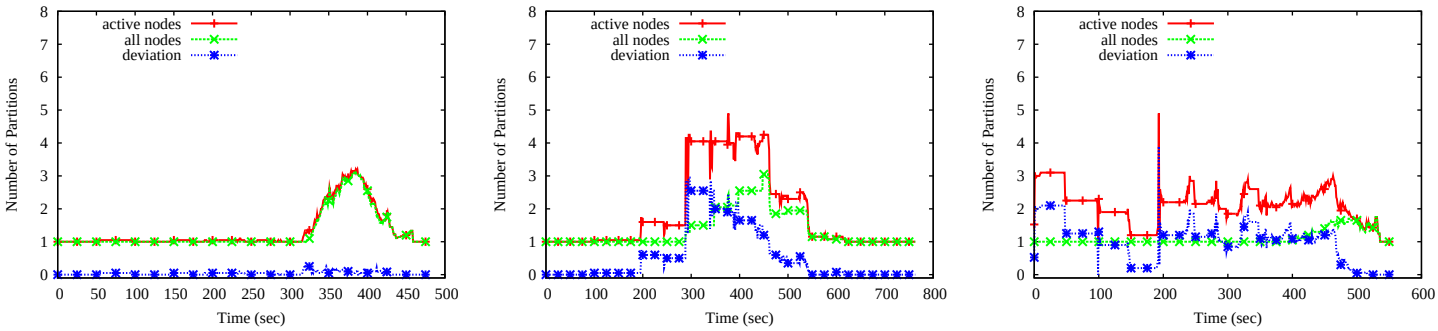
The little peaks in Figure14(a) indicate the iterative selection process of TECA until the topology reaches a stable state. Compared to GAF and ASCENT, TECA needs more iterations as a result of selecting the best nodes regarding link lifetime and loss.

Figure 15 shows the number of network partitions over time if active, i.e., cluster heads and bridges, as well as all nodes that are alive are considered. Also, the difference between both of them is depicted. If we take all nodes into account, the appropriate number of partitions is a lower bound. Guaranteeing connectivity would therefore require that the number of partitions are the same for both cases, i.e., the difference should be zero. As shown in Figure 15(a), TECA maintains network connectivity very well and outperforms GAF and ASCENT, respectively. The topology of active nodes built up by TECA is near the optimum with respect to network partitions almost for the entire simulation time.

Since GAF and ASCENT do not take network connectivity into account explicitly, the network gets partitioned heavily. Also note that the simulation area is very small, with increasing area size, network partitions are even more likely. Of course, a smaller grid size resp. a higher neighbor threshold leads to denser topologies that are more robust against network partitions and traffic loss but would also result in poorer energy savings.

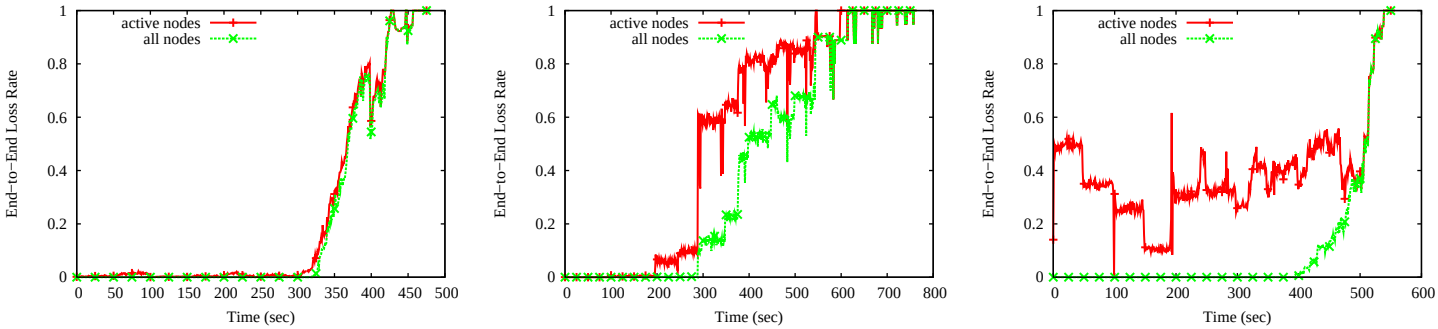
However, short-term partitions sometimes occur even when using TECA because

- it takes some iterations until the topology reaches a steady state,



(a) TECA (b) GAF (c) ASCENT

Fig. 15. Number of network partitions vs. time



(a) TECA (b) GAF (c) ASCENT

Fig. 16. End-to-end loss rate vs. time

- inconsistency in neighbor tables exist due to loss of announcement packets containing state and neighborhood information,
- nodes go to sleep before they know about new clusters due to short passive timeouts and packet loss, or
- link loss estimates are wrong.

Reducing the time until the TECA algorithm terminates would require shorter announcement timers, especially when changes in the neighborhood occur. But the more important issue is the loss of state information. In such cases, nodes might make wrong decisions of becoming active or not. Since all packets are broadcast, transmitting nodes do not get acknowledgements from receiving nodes. Therefore, we propose the use of retransmissions for announcement packets to reduce the probability of packet loss.

Since Figure 15 shows the number of partitions based on 80% connectivity, i.e., nodes are considered connected if the link loss is below LT, wrong loss estimates might cause network partitions, too. For example, consider two cluster heads that assume a link loss of 10% between them based on short-term measurements, although the link quality is actually worse than 80%. Then, adjacent nodes consider both clusters connected and go to sleep even if some of them are needed to connect the clusters.

The end-to-end loss rates over time are depicted in Figure 16. Every second, we randomly select one sender and one receiver among all active nodes. Then, the sending node generates 50 packets and floods the network. However, only active nodes participate in flooding. The selection of sender-receiver pairs is done 50 times.

Based on how many packets get lost at the receiver side, the end-to-end loss rate is calculated. Like in Figure 15, we get a lower bound for the loss rate by taking all nodes into account, i.e., flooding is done by all nodes. Of course, with an increasing number of network partitions, end-to-end loss increases since sender and receiver are selected randomly among all active nodes, independent of their partition.

Again, TECA outperforms GAF and ASCENT. It benefits from keeping network partitions as low as possible. In addition, TECA considers link loss by measuring ongoing packet transmissions. Based on these estimates, bridge nodes are selected to best connect cluster heads. While ASCENT also measures link qualities and considers nodes as neighbors only if the estimated loss rate is below LT, GAF does not take link loss into account at all. It assumes that every node in a grid can communicate with all other nodes in adjacent grids. Therefore, end-to-end loss mainly depends on the grid size used by GAF. A more conservative grid size might lead to better results but would never adapt to dynamic environments with varying link losses. Since ASCENT only considers suitable links with loss rates below LT, it performs better than GAF. However, the algorithm is just based on a neighbor threshold without techniques selecting links that connect two nodes best like TECA.

B. *Investigating Different Node Densities*

The second set of simulations was performed to investigate the impact of node density on the performance of TECA, GAF, and ASCENT. The simulation settings are as in the first simulation set except the node density that we vary between 10 and 50. In addition to the average, all simulation points in the graphs indicate 95% confidence intervals.

First, we are interested in how many nodes are selected to be active. Figure 17 depicts the fraction of active nodes vs. node density at time $t = 20\ s$. As expected, all algorithms benefit from higher node densities expressed by lower active node fractions. Due to the grid constraint in GAF that each node should be able to communicate with any node in adjacent grids, many nodes are activated. Since ASCENT

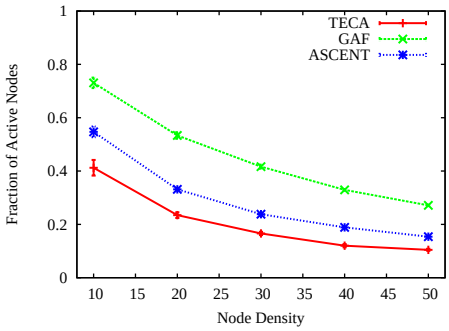


Fig. 17. Fraction of active nodes vs. node density

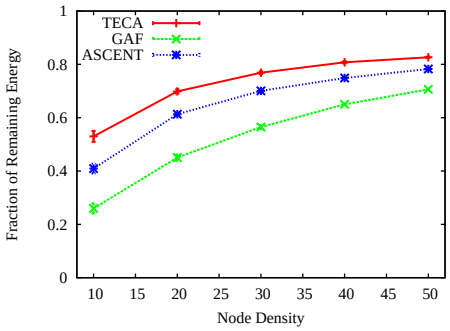


Fig. 18. Remaining energy vs. node density

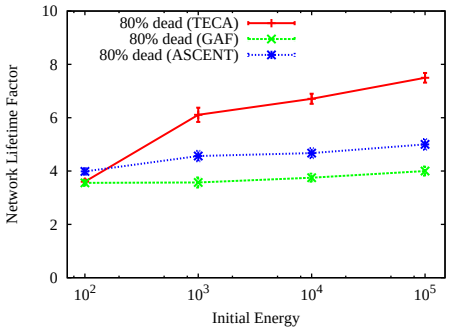
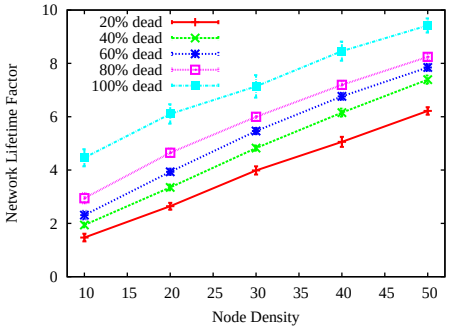
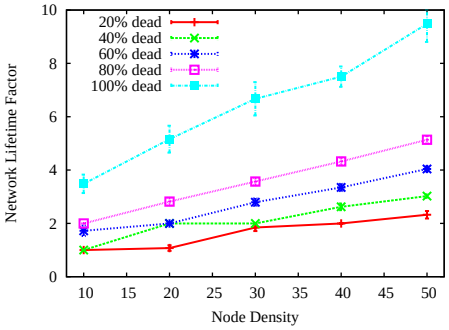


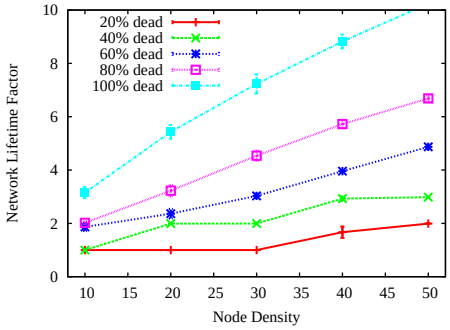
Fig. 19. Network lifetime vs. initial energy



(a) TECA



(b) GAF



(c) ASCENT

Fig. 20. Network lifetime vs. node density

limits the number of active neighbors an active node may have by NT, it performs better. However, TECA outperforms both of them. Due to the fraction of activated nodes, the network's energy consumptions are different. As shown in Figure 18, the fraction of remaining energy averaged over all nodes at time $t = 100\ s$, i.e., after the lifetime of a node that never sleeps, is best for TECA. Because of that, TECA should prolong the network's operational lifetime more than GAF and ASCENT. However, according to Figure 12, GAF performs best until all nodes are dead, followed by ASCENT with TECA performing worst. This is caused by the time nodes stay in passive state with respect to initial energy. In TECA, more iterations are needed than in GAF and ASCENT until the topology is stable. Therefore, more energy is consumed by passive nodes. But with an initial energy value of only $100\ s$, this fraction is crucial for the overall network lifetime.

Figure 19 shows the impact of initial energy on network lifetime for a density of 30 nodes. Varying initial energy from $10^2\ s$ to $10^5\ s$, the network lifetime factor defined by

$$network\ lifetime\ factor = \frac{network\ lifetime}{initial\ energy\ value} \qquad (15)$$

is depicted. Here, network lifetime is the time until 80% of all nodes are dead. With an increasing initial energy value, TECA exploits its full potential. The impact of energy spent by passive nodes on ASCENT and GAF is quite small indicated by a constant network lifetime factor. Thus, for more realistic energy values, TECA is expected to perform best even with respect to overall network lifetime.

The network lifetimes for different fractions of dead nodes vs. node density are shown in Figure 20 (initial energy of $1000\ s$). With TECA, most of the nodes die towards the end of the simulation achieving good energy and load balancing. Since GAF requires a selected node in each grid, grids with few nodes die early. ASCENT employs the worst energy balancing since nodes stay active until they run out of energy.

Note that in GAF the time until all nodes are dead holds out much more than for the case of 80% dead nodes. This is due to the fact that even if most of the nodes are dead, there are still grids with some nodes alive. Although all nodes are distributed over the simulation area uniformly, densely populated grids are not unlikely. Thus, the time until all nodes are dead heavily depends on the maximum number of nodes in a grid.

Figure 21 summarizes the results of TECA, GAF, and ASCENT concerning node density vs. network lifetime. It depicts the network lifetime factor for the case of 80% dead nodes. With an initial energy value of $1000\ s$, TECA achieves the best results, independent of node density.

C. Investigating Different Cluster Timeout Factors

In the last set of simulations, we investigate the impact of different cluster timeout factors on energy balancing and network lifetime. Figure 22 depicts the standard deviation of remaining energy vs. cluster timeout factor for an initial energy value of $1000\ s$ at time $t = 1000\ s$. As expected, different cluster timeouts, i.e., neighbor timeouts at which sleeping nodes will wake up, have no noticeable impact on ASCENT since nodes remain active until they die. However, for GAF and TECA, the standard deviation decreases if nodes wake up more frequently. Although both GAF and TECA rotate the role of being active among all nodes by selecting nodes with most remaining energy, GAF performs worse than TECA since it is bounded by grids. Thus, GAF balances energy consumptions among nodes in the same grid only. Even if just one node is left, the node must be active. In contrast, TECA take all nodes into account leading to a better energy balancing.

At last, Figure 23 presents the influence of cluster timeout factors on network lifetime, again for an initial energy value of $1000\ s$. Unlike before, we now show the network lifetime factor regarding the case of

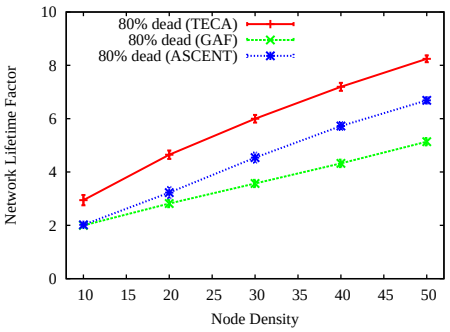


Fig. 21. Network lifetime vs. node density

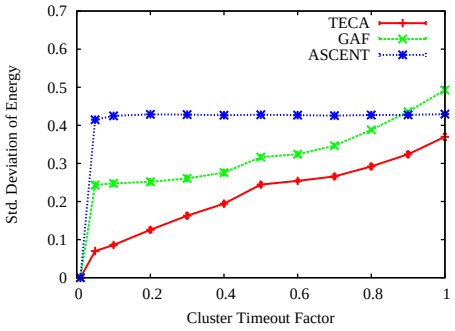


Fig. 22. Standard deviation of remaining energy vs. cluster timeout factor

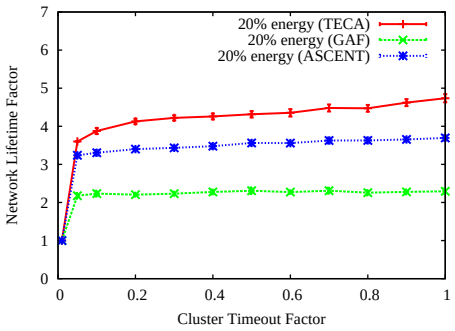


Fig. 23. Network lifetime vs. cluster timeout factor

20% remaining energy in contrast to 80% dead nodes. For all cluster timeout factors, TECA outperforms GAF and ASCENT.

As pointed out above, a significant amount of energy was consumed by passive nodes in TECA. Thus, reducing the wake-up cycle leads to more energy saving since nodes rarely wake up and do not become passive very often. Also, due to the fact that TECA needs several iterations until a stable state is reached, it further benefits from unfrequent cluster timeouts. The impact of increasing cluster timeout factors on GAF and ASCENT is rather insignificant.

VI. Conclusion

In this paper, we have presented a new Topology and Energy Control Algorithm that selects active and sleeping nodes. While active nodes keep their communication radio turned on, sleeping nodes save most energy by turning their radios off. Maintaining network connectivity is an crucial design issue. Based on Minimum Cluster Spanning Trees, TECA works in a distributed and localized fashion being able to guarantee connectivity. By taking packet loss and remaining energy into account, TECA is also able to adapt to different environments and application requirements.

By means of simulations, we have compared TECA to two other approaches, namely GAF and ASCENT. Concerning network connectivity, end-to-end packet delivery, overall network's operational lifetime, and load and energy balancing, TECA outperforms GAF and ASCENT.

Motivated by these promising results, we will extend the simulations by taking asymmetric links as well as MAC layer issues into account. Moreover, we will work on a combination of TECA with adaptive MAC layer protocols that also put active nodes to sleep to reduce idle listening. Thus, active nodes do not need to be active all the time but sleep if they not participate in packet delivery.

Furthermore, we will implement TECA on real sensor nodes. In that way, the simulation results should be verified.

References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A Survey on Sensor Networks. *IEEE Communications Magazine*, 40(8):102–114, August 2002.

[2] L. Bao and J. J. Garcia-Luna-Aceves. Topology Management in Ad Hoc Networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Annapolis, Maryland, June 2003.

[3] D. Blough, M. Leoncini, G. Resta, and P. Santi. The K-Neigh Protocol for Symmetric Topology Control in Ad Hoc Networks. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Annapolis, Maryland, June 2003.

[4] M. Burkhart, P. von Rickenbach, R. Wattenhofer, and A. Zollinger. Does topology control reduce interference? In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Roppongi Hills, Tokyo, May 2004.

[5] A. Cerpa and D. Estrin. ASCENT: Adaptive Self-Configuring sEnsor Networks Topologies. In *Proceedings of IEEE INFOCOM*, New York, USA, June 2002.

[6] A. Cerpa, N.Busek, and D. Estrin. SCALE: A Tool for Simple Connectivity Assessment in Lossy Environments. Technical Report 21, Center for Embedded Networked Sensinsing, UCLA, September 2003.

[7] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. Span: An Energy-Efficient Coordination Algorithm for Topology Maintenance in Ad Hoc Wireless Networks. In *Proceedings of the 7th International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001.

[8] M. B. S. D. E. Culler, D. Estrin. Overview of Sensor Networks. *IEEE Computer Magazine*, 37(8):41–49, August 2004.

[9] O. Dousse, P. Mannersalo, and P. Thiran. Latency of Wireless Sensor Networks with Uncoordinated Power Saving Mechanisms. In *Proceedings of the 5th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Roppongi Hills, Tokyo, May 2004.

[10] A. El-Hoiydi and J.-D. Decotignie. WiseMAC: An Ultra Low Power MAC Protocol for the Downlink of Infrastructure Wireless Sensor Networks. In *Proceedings of 9th IEEE International Symposium on Computers and Communications (ISCC)*, Alexandria, Egypt, June 2004.

[11] P. B. Godfrey and D. Ratajczak. Naps: Scalable, Robust Topology Management in Wireless Ad Hoc Networks. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN)*, Berkeley, California, April 2004.

[12] J. Hill and D. Culler. Mica: A Wireless Platform for Deeply Embedded Networks. *IEEE Mirco*, 22(6):12–24, November 2002.

[13] C. E. Jones, K. M. Sivalingam, P. Agrawal, and J.-C. Chen. A Survey of Energy Efficient Network Protocols for Wireless Networks. *Wireless Networks*, 7(4):343–358, August 2001.

[14] N. Li and J. C. Hou. Topology Control in Heterogeneous Wireless Networks: Problems and Solutions. In *Proceedings of IEEE INFOCOM*, Hong Kong, China, March 2004.

[15] N. Li, J. C. Hou, and L. Sha. Design anf Analysis of an MST-Based Topology Control Algorithm. In *Proceedings of IEEE INFOCOM*, San Francisco, California, April 2003.

[16] N. Nikaein and C. Bonnet. Topology Management for Improving Routing and Network Performances in Mobile Ad Hoc Networks. *Mobile Networks and Applications*, 9(6):583–594, December.

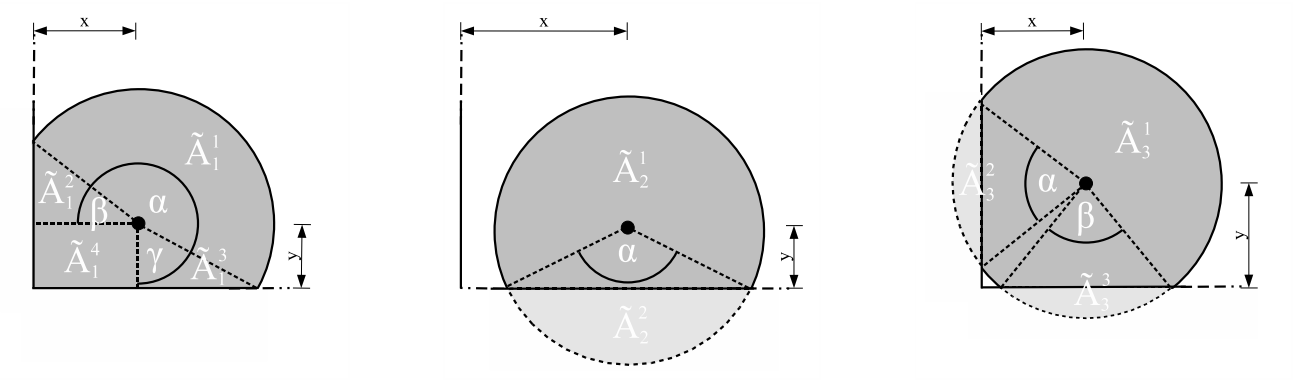
[17] R. Ramanathan and R. Hain. Topology Control of Multihop Wireless Networks Using Transmit Power Adjustment. In *Proceedings of IEEE INFOCOM*, pages 404–413, Tel Aviv, Isreal, March 2000.

[18] J. Schiller, A. Liers, H. Ritter, R. Winter, and T. Voigt. ScatterWeb - Low Power Sensor Nodes and Energy Aware Routing. In *Proceedings of Hawaii International Conference On System Sciences (HICSS)*, Hawaii, USA, January 2005.

[19] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Optimizing Sensor Networks in the Energy-Latency-Density Design Space. *IEEE Transactions on Mobile Computing*, 1(1):70–80, January 2002.

[20] C. Schurgers, V. Tsiatsis, S. Ganeriwal, and M. Srivastava. Topology Management for Sensor Networks: Exploiting Latency and Density. In *Proceedings of the 3rd ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, Lausanne, Switzerland, June 2002.

[21] T. van Dam and K. Langendoen. An Adaptive Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of ACM SenSys*, Los Angeles, CA, November 2003.



(a) Case 1 (b) Case 2 (c) Case 3

Fig. 24. Covered communication area by simulation area

[22] Y. Xu, S. Bien, Y. Mori, J. Heidemann, and D. Estrin. Topology Control to Conserve Energy in Wireless Ad Hoc Networks. Technical Report 6, Center for Embedded Networked Sensinsing, UCLA, January 2003. Submitted for review to IEEE Transactions on Mobile Computing.

[23] Y. Xu, J. Heidemann, and D. Estrin. Adaptive Energy-Conserving Routing for Multihop Ad Hoc Networks. Technical Report 527, Information Sciences Institute, USC, October 2000. Submitted for publication.

[24] Y. Xu, J. Heidemann, and D. Estrin. Geography-Informed Energy Conservation for Ad Hoc Routing. In *Proceedings of the 7th International Conference on Mobile Computing and Networking (MobiCom)*, Rome, Italy, July 2001.

[25] W. Ye, J. Heidemann, and D. Estrin. An Energy-Efficient MAC Protocol for Wireless Sensor Networks. In *Proceedings of IEEE INFOCOM*, New York, NY, June 2002.

[26] M. Zuniga and B. Krishnamachari. Analyzing the Transitional Region in Low Power Wireless Links. In *Proceedings of IEEE Secon*, Santa Clara, CA, October 2004.

APPENDIX I
DERIVATION OF EQUATION 11

Consider an area of size $A \times A$ with $A > 0$. Let N be the number of nodes placed on the area. Let p be the probability that a node is linked to an adjacent neighbor. Then, the probability of having k neighbors is given by

$$P(X = k) = \binom{N-1}{k} p^k (1-p)^{N-k-1}. \tag{16}$$

Let μ be the node density, i.e., the average number of nodes covered by a node's communication range. Then, we get $\mu = (N-1)p + 1$ that is equivalent to

$$N = \frac{\mu - 1}{p} + 1. \tag{17}$$

Thus, it remains to find probability p. Let $\langle x, y \rangle$ be the node's position and $r < \frac{1}{2}A$ be the radius of its communication area. Then, four cases can be distinguished:

1) *The node's location is in one of the four corners.*
Let $\tilde{A}_1$ be the area of the node's communication range restricted by the simulation area. As shown in Figure 24(a), $\tilde{A}_1$ is composed of four subareas:

$$\begin{aligned} \tilde{A}_1 &= \tilde{A}_1^1 + \tilde{A}_1^2 + \tilde{A}_1^3 + \tilde{A}_1^4 \\ &= \frac{1}{2}\alpha r^2 + \frac{1}{2} rx \sin\beta + \frac{1}{2} ry \sin\gamma + xy \end{aligned} \tag{18}$$

with $\alpha + \beta + \gamma = \frac{3}{2}\pi$, $\beta = \cos^{-1}\frac{x}{r}$, and $\gamma = \cos^{-1}\frac{y}{r}$. Then, the average area $\bar{A}_1$ is given by

$$\bar{A}_1 = \frac{1}{\frac{1}{4}\pi r^2} \int_0^r \int_0^{\sqrt{r^2-y^2}} \tilde{A}_1 \, dx \, dy \tag{19}$$

2) *The node's location is at only one boundary.*
According to Figure 24(b), $\tilde{A}_2$ is calculated by

$$\begin{aligned} \tilde{A}_2 &= (\tilde{A}_2^1 + \tilde{A}_2^2) - \tilde{A}_2^2 \\ &= \pi r^2 - \frac{r^2}{2}(\alpha - \sin\alpha) \end{aligned} \tag{20}$$

with $\alpha = 2\cos^{-1}\frac{y}{r}$. The average area $\bar{A}_2$ is then

$$\begin{aligned} \bar{A}_2 &= \frac{1}{r(A-2r)} \int_0^r \int_r^{A-r} \tilde{A}_2 \, dx \, dy \\ &= r^2(\pi - \frac{2}{3}). \end{aligned} \tag{21}$$

3) *The node's location is at both boundaries, but not in a corner.*
Analog to the second case, area $\tilde{A}_3$ is given by

$$\begin{aligned} \tilde{A}_3 &= (\tilde{A}_3^1 + \tilde{A}_3^2 + \tilde{A}_3^3) - \tilde{A}_3^2 - \tilde{A}_3^3 \\ &= \pi r^2 - \frac{r^2}{2}(\alpha - \sin\alpha) - \frac{r^2}{2}(\beta - \sin\beta) \end{aligned} \tag{22}$$

with $\alpha = 2\cos^{-1}\frac{y}{r}$ and $\beta = 2\cos^{-1}\frac{x}{r}$, as depicted in Figure 24(c). The average area $\bar{A}_3$ is

$$\bar{A}_3 = \frac{1}{r^2(1 - \frac{1}{4}\pi)} \int_0^r \int_{\sqrt{r^2-y^2}}^r \tilde{A}_3 \, dx \, dy. \tag{23}$$

4) *The node's communication area is not restricted by the simulation area.*
In this case, we just get

$$\bar{A}_4 = \tilde{A}_4 = \pi r^2. \tag{24}$$

Let $\bar{p}_1$, $\bar{p}_2$, $\bar{p}_3$, and $\bar{p}_4$ be the probabilities of the different cases. Then, the average covered communication area $E[A]$ of a node arbitrarily placed on the simulation area is given by

$$E[A] = \sum_{i=1}^{4} \bar{p}_i \bar{A}_i \tag{25}$$

with $\bar{p}_1 = \frac{\pi r^2}{A^2}$, $\bar{p}_2 = \frac{4r(A-2r)}{A^2}$, $\bar{p}_3 = \frac{r^2(4-\pi)}{A^2}$, and $\bar{p}_4 = (1 - \bar{p}_1 - \bar{p}_2 - \bar{p}_3)$. Thus, the probability p is defined by

$$p = \frac{E[A]}{A^2} \approx \frac{r^2(3.142A^2 - 2.667Ar + 0.158r^2)}{A^4}. \tag{26}$$