

# A Distributed Algorithm for Hexagonal Topology Formation in Wireless Sensor Networks \*

K. Shashi Prabh<sup>†</sup>

Chinmay Deshmukh<sup>‡</sup>

Shikhar Sachan<sup>‡</sup>

## Abstract

*Hexagonal wireless sensor network refers to a network topology where a subset of nodes have six peer neighbors. These nodes form a backbone for multi-hop communications. In a previous work, we proposed the use of hexagonal topology in wireless sensor networks and discussed its properties in relation to real-time (bounded latency) multi-hop communications in large-scale deployments. In that work, we did not consider the problem of hexagonal topology formation in practice – which is the subject of this research. In this paper, we present a decentralized algorithm that forms the hexagonal topology backbone in an arbitrary but sufficiently dense network deployment. We implemented a prototype of our algorithm in NesC for TinyOS based platforms. We present data from field tests of our implementation, collected using a deployment of fifty wireless sensor nodes.*

## 1 Introduction

Wireless Sensor Networks (WSN) refers to ad-hoc wireless networks consisting of nodes that have sensing capability. The distributed sensed data is collected and processed by a subset of nodes, that generally involves moving data over multiple hops. Since the channel is shared and interference can cause corrupted packet receptions, its access needs to be regulated so as to provide bounded latency guarantees. The end-to-end latency of such communications must be bounded to enable real-time applications over WSN, which are important as enabling technology for novel applications, such as, healthcare through smart nursing homes, monitoring of critical infrastructure and real-time tracking. However, providing bandwidth guarantees in wireless ad-hoc networks has been proven to be NP-Hard [1].

There exist many TDMA based solutions that sched-

ule transmissions using message exchanges (to reach an agreement) among neighboring nodes [12, 13, 14]. This mechanism is communication intensive and often carries significant processing overhead. Although this overhead is not a major issue in traditional networks, such is not the case for WSN where nodes have limited energy supply. An alternative to this approach, proposed in [10], considers the formation of a logical regular topology backbone for multi-hop routing of packets. It was shown that for hexagonal node connectivity, one can obtain closed-form expressions that determine TDMA schedules for convergecast. Thus, not only TDMA scheduling costs nearly zero communication overhead, but also one can determine the guaranteed bandwidth and the worst-case end-to-end delay of packets. Considering regular, instead of arbitrary, topology eliminates the NP-Hardness issues of bandwidth guarantees. Not only that, due to the regularity of topology, it affords simpler and more efficient networking protocols.

By *hexagonal topology*, we mean a network topology where the nodes have six neighbors, except for those that are at the edges of the network. This refers to node connectivity, and is in contrast to *hexagonal tessellation* which is often used to model cellular networks. In the latter, one tessellates, or divides, the geographical area into hexagonally shaped cells. Strictly speaking, hexagonal topology is equivalent to equilateral triangular tessellation (Figure 1). The corners of the triangles are sites of nodes, which are called hexagonal (also, equilateral triangular) lattice points. Due to practical reasons, which we discuss below, one can only expect to realize hexagonal topology whose node locations only approximate the hexagonal lattice points in typical WSN deployments.

Forming hexagonal topology in WSNs presents some unique difficulties. WSNs are characterized by large number of nodes. Networks with hundreds of nodes are not uncommon today. Some researchers have built WSNs with thousands of nodes. Therefore, nodes can't be assumed to be placed at hand-picked locations. Physical features of the deployment area may also exclude certain locations from node deployment. Furthermore, since radio range varies with space and time, it is not possible to form specific topologies by the means of physical placement considerations alone. The work presented in this paper maintains a lower bound on the separation of neighboring

\*This work was supported by Fundação para a Ciência e a Tecnologia, Portugal.

<sup>†</sup>K. S. Prabh is with Centro de Investigação em Sistemas de Tempo-Real, Instituto Superior de Engenharia do Porto, Portugal. email: ksp@isep.ipp.pt

<sup>‡</sup>C. Deshmukh and S. Sachan are with Department of Computer Science, Indian Institute of Technology, Guwahati, India. email: {chinmay,s.sachan}@iitg.ernet.in

nodes of hexagonal topology. In other words, the resulting topology is *semi-logical* hexagonal.

In this paper, we present a distributed algorithm to form semi-logical hexagonal topology in WSN deployments. We implemented this algorithm in NesC. We also wrote a simulator for the algorithm in C++. We describe the prototype implementation and present data obtained from deployments of TelosB and MicaZ motes.

The rest of this paper is structured as follows: In Section 2, we present related work. We present the algorithm to form hexagonal WSN in Section 3, followed by a simulation result in Section 4. We present a discussion of our implementation of the algorithm in Section 5. We present conclusions in Section 6.

## 2 Related Work

Takagi and Kleinrock established that the progress of a multi-hop packet in wireless networks is maximized when each node has 8 neighbors – the so called “magic number” [17]. This result inspired the design of K-neigh topology control algorithm [3]. The K-neigh topology control algorithm forms a topology with approximately  $k$  neighbors to each node. COMPOW forms connected topology at a common minimum transmission power level [9]. Xue and Kumar established that for a flat (all nodes are peers) network of  $n$  nodes to be fully connected, the number of neighbors grow as  $\Theta(\log n)$  [18]. This result indicates that a flat network organization is unsuitable for large-scale WSN.

In a backbone based network (the approach taken in this paper), a dominating set of nodes are used to route multi-hop packets. The nodes that are not in this set are one hop from some node in the dominating set. Clearly, it is desirable that the cardinality of the dominating set be as small as possible, but this is an NP-Complete problem [6]. SPAN is a heuristic that finds an approximate dominating set [4]. In a hexagonal WSN, the backbone nodes form a dominating set where the elements of this set are constrained to have six peer neighbors.

Previous topology control research has typically focused on achieving a certain node degree, ensuring connectivity, maintaining certain throughput etc. [11, 15], but not on regular topology formation techniques. In a previous work [10], we introduced hexagonal wireless sensor network and discussed its properties in relation to the problem of bounded latency multi-hop communications in large-scale sensor networks. In that work, we did not consider the problem of hexagonal topology formation in practice. In this paper, we present a decentralized algorithm that forms the hexagonal topology backbone. To the best of our knowledge, no previous implementation of hexagonal or any other regular topology formation in the wireless domain exists. We note that a system of (wired) multiprocessors arranged in hexagonal topology has been reported previously [5, 16].

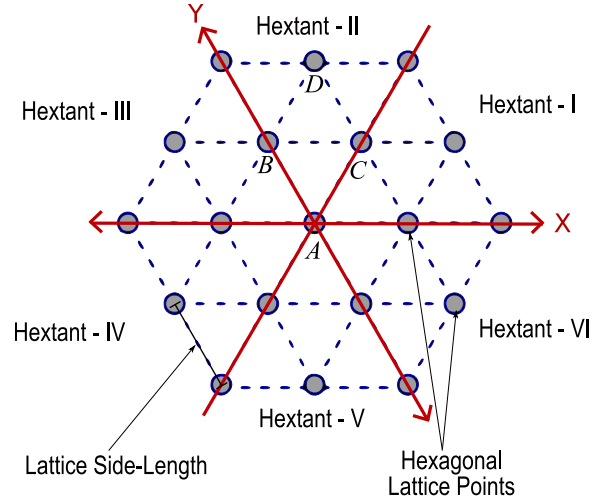


Figure 1. Hexagonal Topology

## 3 Semi-Logical Hexagonal Topology Algorithm

In this section, we present an algorithm that selects, from an arbitrary node deployment, a subset of nodes connected in hexagonal topology. The physical placements of the hexagonal nodes are *weakly correlated* to hexagonal lattice points. In other words, the deviation of actual location of hexagonal nodes from ideal hexagonal lattice points is bounded, or, the resulting hexagonal topology is semi-logical. A hexagonal lattice is completely specified by only one pair of neighboring lattice points (1). The euclidean distance between two neighboring lattice points is called the *lattice side-length*, or, side-length for short (Figure 1). In our algorithm, the target side-length is an input parameter.

The nodes of hexagonal WSN are selected from a circular area centered at the ideal lattice points, which are referred to as *lattice sites* henceforth. We chose circular regions for the simplicity of computation. The radius of lattice sites is also an input parameter. Errors of localization need to be factored-in to determine the size of lattice sites. We denote the side-length of hexagonal lattice by  $S$  and the radius of lattice sites by  $\sigma$ . Observe that the distance between any two neighboring nodes is upper bounded by  $S + 2\sigma$ . We denote the minimum power level of node  $n_i$ 's radio to transmit to an arbitrary node at distance  $S + 2\sigma$  away by  $P_i(S + 2\sigma)$ .

The hexagonal lattice has three main diagonals (Figure 1). We select a pair of the diagonals inclined at  $120^\circ$  to be the  $X$  and  $Y$  axes. The three diagonals divide the lattice into six symmetric regions, which we call *hexants*. Out of the two diagonals that define a hexant, we adopt the convention of including the first anti-clockwise diagonal and the lattice sites centered on this diagonal to the hexant. For example, in Figure 1, lattice sites  $C$  and  $D$  are defined to be in the second hexant of  $A$  while the site  $B$  is defined to be in the third hexant of  $A$ . We also say that  $C$  is a first hop neighbor lattice site of  $A$  in the latter's

second hextant.

We denote a node by  $n_i$  and its Cartesian coordinates by  $(x_i, y_i)$ . We assume that the nodes are localized prior to running this algorithm [2, 7]. The process of topology formation is initiated by a special node,  $n_0$  located at  $(x_0, y_0)$ . Another coordinate pair at distance  $S$  from  $n_0$ ,  $(x_1, y_1)$ , defines the  $X$ -axis and lattice side-length. The hexagonal lattice points are given by

$$x_0 \pm nS/2, y_0 \pm m\sqrt{3}S/2, \quad (1)$$

where  $n$  and  $m$  are integers. In practice, one may program the initializer node with the special node ID  $n_0$ . Upon the completion of node localization process,  $n_0$  will start the hexagonal topology formation. To deal with node failures or dynamic changes in the network, either  $n_0$  may be programmed to trigger the hexagonal topology formation process periodically or the process of peer neighbor selection may be run locally.

Our algorithm for hexagonal topology formation is presented in Figures 2–3, except for some optimizations and non-essential details. In this algorithm, a node needs to distinguish its neighbors (and the initializer node  $n_0$ ). However, one can't always assume the existence of unique node addresses in WSN deployments. The word "ID"s in the following presentation should be interpreted in a more general sense. A unique node ID may be derived from the node's location, for example.

First, node  $n_0$  selects its hexagonal neighbors. The newly selected hexagonal nodes then select their peer neighbors and so on. After  $n_0$  finishes, multiple nodes engage in topology formation in parallel. The algorithm is described in detail below. Paragraphs that begin with "[Lines x.y–z]" describe lines y to z of the algorithm presented in Figure x.

[Lines 2.4 – 9]. First, each node builds a table of the locations of its neighbors. This neighbor table construction step is initiated by node  $n_0$  that broadcasts an `InitPacket`. Besides the sender node's identification, these packets contain  $\{x_0, y_0; x_1, y_1; \sigma; x_i, y_i\}$  where  $(x_i, y_i)$  are coordinates of the sender node. Upon receiving an `InitPacket` for the first time, nodes determine the Euclidean distance between their location and the nearest hexagonal lattice point using (1). If this distance is larger than  $\sigma$ , then the node exits the topology formation or, "drops-out," otherwise, it broadcasts an `InitPacket` and inserts the sender's ID and location in its neighbor table. Upon receiving subsequent `InitPackets`, only the neighbor table is updated.

[Lines 2.11–13]. These steps start at  $n_0$  after sufficient time interval to allow the initialization broadcasts to reach nodes 3 hops away. A hexagonal node selects its neighbors from a set of potential candidates by applying a series of evaluation criteria (quality indicating metrics). This is accomplished by collecting neighbors' neighbor tables, count of non-empty neighbor lattice sites and count of links with non-neighbor lattice sites. The last one is split into two – count of links with non-neighbor lattice-

## SEMILOGICAL-HEXAGONAL-TOPOLOGY ( $S, \sigma$ )

▷ This algorithm finds backbone nodes in the neighborhood of hexagonal lattice points.

INITIALIZATION:

```

1 IsHexNode ( $n_0$ ) ← TRUE
2 IsHexNode ( $n_i$ ) ← FALSE  $\forall i \neq 0$ 
3 HexPeerSelectionDone ( $n_i$ ) ← FALSE  $\forall i$ 

▷ InitPacket from node  $n_i$  contains
   $\{x_0, y_0; x_1, y_1; \sigma; x_i, y_i\}$ 
4  $n_0$  broadcasts InitPacket at Tx power  $P_0 (S + 2\sigma)$ 
5 do Upon the first reception of InitPacket
6   if This node's location is within distance
      $\sigma$  from some lattice point
     then
7     Broadcast InitPacket ( $n_i$ ) at power  $P_i (S + 2\sigma)$ 
8     Update neighbor table using InitPacket
     else ▷ Drop out
9     EXIT
     endif
   enddo
10 Use subsequent InitPackets to augment neighbor table

▷ Out of the two diagonals that define a hextant, we adopt
the convention of including the first anti-clockwise diagonal
and the lattice sites centered on this diagonal to the hextant.
▷  $N_i[18]$  represents an array of node IDs. For  $k = 1 \dots 6$ ,
 $N_i[k] \neq \text{UNSET}$  indicates that some neighbor of  $n_i$  in
hextant  $k$  has been selected. Similarly, for  $k = 7 \dots 18$ ,
 $N_i[k]$  indicate the second hop neighbors.
▷ "Non-empty lattice sites"  $\Rightarrow$  sites with more than 0 nodes.
11 if  $\{\text{IsHexNode } (n_i)\}$  AND  $\{\neg \text{HexPeerSelectionDone } (n_i)\}$ 
    AND  $\{\text{NoRaceCondition } (n_i)\}$ 
    then
12   Request for neighbors' neighbor tables, the number
    of non-empty neighbor lattice sites and long-links
13   Call SELECT-BEST-NEIGHBORS to select
    new neighbors in hextants  $N_i[1..6] = \text{UNSET}$ 
14   Update and broadcast  $N_i[18]$  in SelectPacket
15   HexPeerSelectionDone ( $n_i$ ) ← TRUE
    endif
17 do Upon receiving SelectPacket
18   if This node's ID  $n_i \in N_j[18]$ 
19     then
20     IsHexNode ( $n_i$ ) ← TRUE
21   elseif This node's location is within
    distance  $\sigma$  from some  $n_k \in N_j[18]$ 
    then ▷ Drop out
22   EXIT
    endif
  enddo

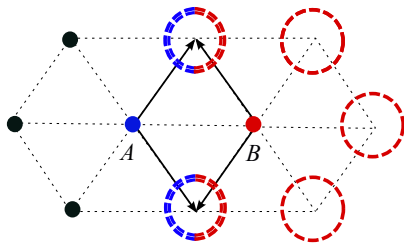
```

Figure 2. Semi-logical hexagonal topology formation algorithm.

### SELECT-BEST-NEIGHBORS

- ▷ Node  $n_i$  maintains an array  $p[6]$ . Each response of the broadcast request (of a hexagonal node to select its new peers) contains the count of a node's non-empty neighbor lattice sites.  $p[k]$  contains the maximum of non-empty neighbor lattice site count of all responses from nodes at the lattice site in hextant  $k$ .
- 1 **do** Upon receipt of the response from each node  $n_j$ 
  - ▷ Let  $q(n_j)$  be the count of non-empty neighbor lattice sites of  $n_j$ . Let  $k$  be the hextant of  $n_j$  w.r.t.  $n_i$ .
  - 2 **if**  $p[k] > q(n_j)$  OR  $N[k] \neq \text{UNSET}$ 
    - 3 **then**
      - Purge the response
    - 4 **else**
      - Update  $p[k]$  if necessary
      - Purge other responses that have non-empty neighbor lattice site count  $< p[k]$
  - 5 **endif**
- 6 **enddo**
- ▷ After all responses are received (or at time-out):
- 7 **if**  $p[i]$ 
  - 8 **then**
    - Remove responses of nodes from site  $i$  that don't have neighbors at the sites  $(i \pm 1) \% 6$ , unless the latter are empty (If  $p[(i \pm 1) \% 6]$  OR  $N[(i \pm 1) \% 6]$  is false then the nodes at site  $i$  are at the edge.)
  - 9 **endif**
- ▷ Now, all nodes in the response set are viable and in a certain sense optimal candidates (See Claim 1 for a proof).
- 10 Sort the remaining candidates in increasing  $LLF$  where  $LLF$  is the count of previously selected nodes that are more than 1 hop away from the candidate node but are in its neighbor table.
- 11 If the number of candidates with the smallest  $LLF$  is more than one, further sort these in increasing  $LLU$ , where  $LLU$  is the count of lattice sites that are more than 1 hop away from the candidate node but at-least one node of such sites are in its neighbor table.
- 12 If the number of candidates with the smallest  $LLU$  and  $LLF$  is more than one, sort these in the order of decreasing LQI, where LQI stands for the link quality indicator. Pick the node with the largest LQI.

**Figure 3. Hexagonal neighbor selection.**



**Figure 4. Common lattice sites of A and B**

sites where hexagonal nodes are already selected and that where hexagonal nodes are yet to be selected. After broadcasting the request, SELECT-BEST-NEIGHBORS is called to process the responses. We associate an array  $N$  of 18 elements with each hexagonal node (or, each lattice site). The elements of  $N$  contain the Node-IDs of its 6 first-hop neighbors and 12 second-hop neighbors in a specified order. In our implementation, we chose anti-clockwise order starting at the x-axis. The unset elements of  $N$  indicate the sites where peer nodes are yet to be selected. Routine SELECT-BEST-NEIGHBORS applies a set of metrics to select its peer neighbors.

[Lines 3.1–5]. The count of a node's neighboring lattice sites that are non-empty is used as the most significant metric. The goal of this metric is to favor those nodes that offer extending the topology to larger extent when they will be selecting their peers (i.e., to increase the geographical coverage of the network with hexagonal nodes). Observe that this is only a local optimization.

[Line 3.7]. In the following, we use the lower-case letters  $a$  and  $b$  for nodes, and the upper-case letters  $A$  and  $B$  for the lattice sites that contain them. Let some node  $a$  be selecting its peers (Figure 4). If node  $b$  is selected, then it should also be the neighbor of two hexagonal nodes at the common neighborhood sites of  $a$  and  $b$ , which we denote by  $AB_{\pm}$ . Node  $a$  can determine whether  $b$  is a neighbor of some already selected hexagonal node (by  $a$  or by some other node earlier) at these common sites by scanning the neighbor table of  $b$ . Thus,  $b$  sends the list of its neighbors at the sites  $AB_{\pm}$  (only). Node  $a$  retains responses of only those nodes at the lattice site  $B$  that have neighbors at the lattice sites  $AB_{\pm}$  (unless the sites  $AB_{\pm}$  are empty, which may happen if the nodes are at the edge) while it discards others.

**Claim 1.** Suppose that in the neighborhood of node  $a$ ,  $\alpha_i$  is the set of peer candidate nodes at lattice site  $A_i$  who have neighbors at lattice sites  $A_{i \pm 1 \% 6}$  whenever  $A_{i \pm 1 \% 6}$  are non-empty. Let  $q(a)$  be the number of the peer neighbors of  $a$ . Then, any greedy selection of peer nodes from set  $\alpha_i$  maximizes  $q(a)$ .

*Proof.* Proof by contradiction. Since hexagonal nodes are selected by other peer nodes, there exists at-least one neighbor of  $a$  (except when  $a$  is the base node  $n_o$ , where an empty peer set makes this claim trivially true). Let us assume that the greedy selection finds a neighbor  $b$  in hextant  $k$  but not in at-least one of the neighboring hex-tants of  $k$ , say  $l$ , whereas, some other selection process finds a neighbor  $b'$  in hextant  $k$  and  $l$ . We have a contradiction if  $b$  and  $b'$  are the same. Otherwise,  $b'$  must have been available to the greedy selection, since only those nodes are discarded from  $\alpha_i$  that have no neighbors in their common lattice sites. Since  $b'$  was available to the greedy selection,  $b$ , which has no neighbor in hex-tant  $l$  must have been discarded. Once again, we have a contradiction.  $\square$

[Lines 3.8–10]. If more than one candidate for hexagonal node exist at some given lattice site, then the following metrics are applied in sequence (in decreasing order of importance):

- Nodes that have the minimum number of long-links with previously selected hexagonal nodes. We chose this filtering step as the most important one since it has the effect of favoring those nodes whose transmissions interfere with the least number of other *backbone* nodes.
- Nodes that have the minimum number of long-links with some node in lattice sites that are more than 1-hop away. This filtering step has the effect of eliminating those nodes whose transmission range is too large in order to mitigate interference.
- Nodes that offers the best link quality.

[Lines 2.14–22]. Once a node completes the neighbor selection process, it broadcasts its list of 1 and 2-hop neighbors. Upon selection of some hexagonal node at any given lattice site, all other nodes at this site exit the topology selection process.

**Race avoidance.** Race conditions may arise when newly selected nodes start the process of selecting their peers in parallel (Line 2.11). For example, in Figure 1, some node at site *A* selects new nodes at *B* and *C*. Since site *D* is common to *B* and *C*, the process of peer node selection by *B* and *C* must be serialized.

For race avoidance, the newly selected nodes send a `ReadyToSelect` advertisement broadcast before proceeding to initiate the process of selection of their peers. Nodes that are already in the process of peer selection send `DenySelectionInitialization` messages. Otherwise, a `OKSelectionInitialization` message is sent. Ties are broken in favor of smaller Node ID. If a node receives any `DenySelectionInitialization` message, it suspends the selection process till it hears the broadcast done at the end of a selection process or a time-out. If a node has received `OKSelectionInitialization` contention messages from all nodes that may cause the race condition or, if it does not receive any `DenySelectionInitialization` message until a time-out, it starts the process of selecting its peers (Line 2.11). This race avoidance mechanism may fail if the envelop of the topology formation process becomes concave (Placing  $n_0$  at the center of the deployment helps in this respect).

**Termination of the topology formation.** When some newly selected node can't find a new peer node, it concludes that it is at the network's edge and hence sends topology termination message to the node that selected it. When a node receives termination message from all the nodes that it selected, it sends its own termination message. The topology formation process terminates when all the children of node  $n_0$  terminate.

## 4 Simulation

We wrote a simulator for our hexagonal topology formation algorithm.<sup>1</sup> We present the result of a run of our algorithm on a network of 100 nodes arranged in 25 hexagonal lattice sites. The sites were arranged in 5 rows of 5 sites each. The radius of lattice sites was  $0.4 * side\ length$ . Within each lattice site, we distributed 4 nodes at uniformly random locations. We added random fluctuations to the radio signal strength. Figure 5 shows the connectivity graph of a flat network (all nodes are peers). Figure 6 shows the simulator generated hexagonal topology on the same graph. Nodes connected by solid (red) lines form the hexagonal backbone.

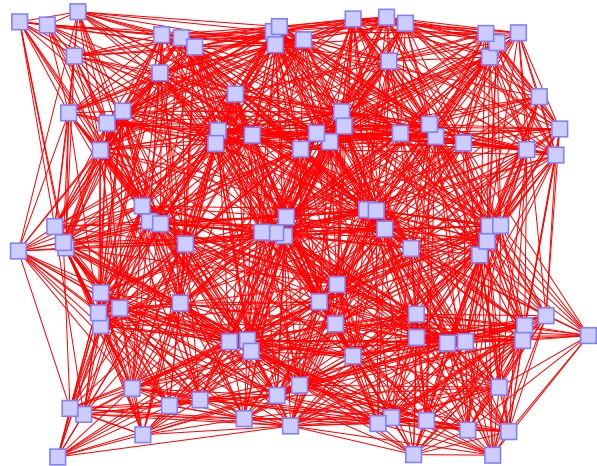


Figure 5. Connectivity graph: Initially

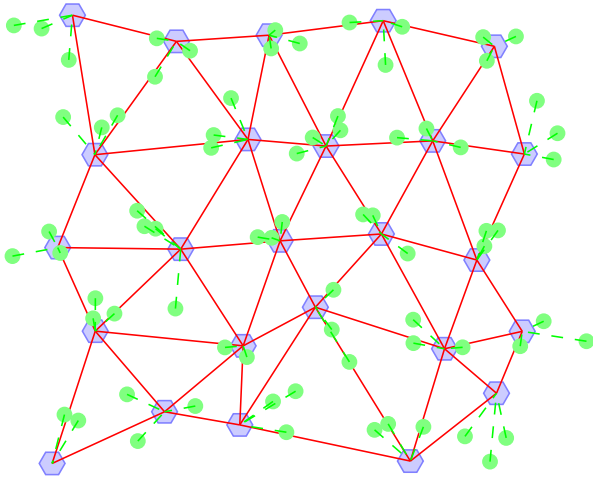
## 5 Prototype Implementation

We implemented our hexagonal topology formation algorithm in NesC for platforms running TinyOS and conducted field experiments with this implementation to establish the feasibility of decentralized hexagonal topology formation in WSN deployments. In this section, we present the details of our implementation and the experiments.

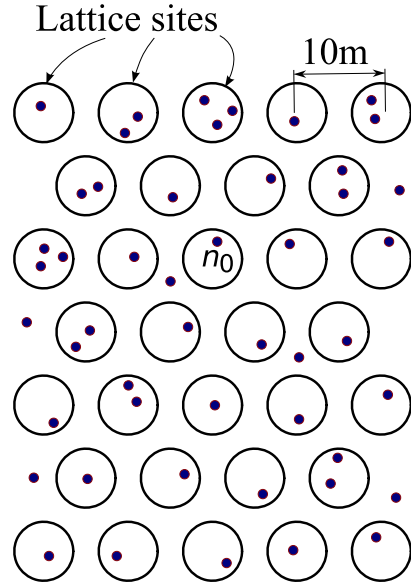
We used TelosB and MicaZ motes. TelosB motes feature MSP430 micro-controller, 10KB volatile memory, 48KB program memory and integrated light, temperature and humidity sensors. MicaZ motes feature MPR2400 micro-controller, 4KB volatile memory and 128KB program memory, and sensor boards are attached through an expansion slot. Both platforms use CC2420 radio transceiver, which has programmable transmission power level. The CC2420 transceiver provides received signal strength indicator (RSSI) and link quality indicator (LQI) measurements of each received packet.

TelosB motes have internal monopole antenna. We observed that successful packet transmission between a pair

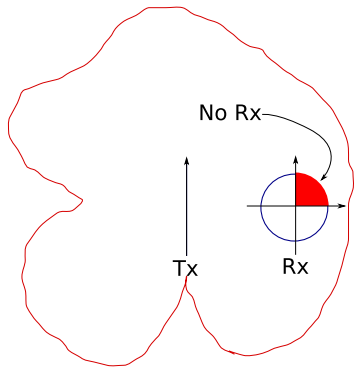
<sup>1</sup>We are putting the simulator code in the public domain. We have not provided the URL here since it may change. The code can also be obtained by contacting the first author.



**Figure 6. Connectivity graph: After running hexagonal topology formation algorithm**



**Figure 8. Deployment layout**



**Figure 7. Dead-zones of communication between a pair of TelosB motes**

of TelosB motes not only depends on the distance between the nodes but also on their *relative orientation*. Certain placement configurations when these motes can't communicate even if the radiated power of the transmitter at the receiver's location should be sufficiently high, span almost  $90^\circ$ . Other configurations where these dead-zones extend to very small angular span also exist. In Figure 7, we show one placement configuration where dead-zone extends to  $90^\circ$ . The arrows correspond to the line that passes through the center of the mote and the arrowhead aligns with the location of the mote's USB port. Replacing the internal antenna of TelosB with an external antenna solves this problem.

We deployed about 50 motes in a  $50 \text{ ft} \times 120 \text{ ft}$  auditorium. The schematics of the deployment is shown in Figure 8. Our target was a hexagonal backbone of 32 nodes arranged in 7 rows where each row was either 4 or 5 hops wide. The target side-length of hexagonal topology was 10 ft. The diameter of the network was 7 hops or, approximately 70 ft. The topology formation initializer node,  $n_0$

was placed in the center of the fifth row. Localization information was added manually before compiling and loading program images to the motes. The motes were placed at approximately the pre-specified coordinates.

Since some of the steps of topology formation algorithm start upon receiving of broadcast messages, a number of nodes can execute in synchrony. If packet transmissions follow the broadcast, hidden node collisions may occur. To reduce such collisions, we added random waits after receiving such broadcasts. Furthermore, to ensure the reliability of the critical transmissions, we used packet acknowledgments. However, TelosB motes generates false acknowledgments frequently (This issue is reported in TinyOS Enhancement Proposal 127 [8].) We used software generated acknowledgments as a workaround.

The selection of the radius of the lattice site involves a trade-off between node density and the extent of interference. It is immediately clear that as the area of lattice sites become smaller, a larger density of nodes is needed to ensure that the lattice sites are not empty. In Section 3, we pointed out that the desirable transmission range of hexagonal backbone nodes is  $side-length + 2\sigma$ . Since interference range is larger than the transmission range, larger  $\sigma$  implies that the number of hexagonal nodes interfered by a given transmission increases with  $\sigma$ . Furthermore, the distance between any two neighboring hexagonal nodes is lower bounded by  $side-length - 2\sigma$ . Therefore,  $\sigma \leq side-length/2$  to ensure that no node belongs to more than one lattice site. In dense deployments, large values of  $\sigma$  lead to large number of messages in response of neighbor selection broadcast (Line 3.1). It may lead problems due to limited memory capacity of the nodes. In such cases, a suitably small value for  $\sigma$  needs to be chosen. Finally, selecting a too small  $\sigma$  may leave holes in the network.

We chose the radius of the lattice sites as  $side-length/3$

or, 3.3 ft. We ensured that at-least one node was placed in each lattice site. As we explained in Section 3, to speed-up the topology formation process, a subset of newly selected hexagonal nodes execute the process of the selection of peer nodes in parallel (Line 13, Figure 2). It is necessary to avoid the race conditions that arise due to this parallelization. To test the race avoidance properties of our implementation, we placed multiple nodes at the locations where we anticipated race conditions. We repeated the experiment 10 times. In all repetitions, one and only one node from each lattice site was selected.

The average time to complete the formation of hexagonal topology was 78 seconds. The minimum time taken to complete the topology formation was 50 seconds and the maximum was 108 seconds. The focus of this experimental evaluation was to establish the feasibility of hexagonal WSN on existing hardware. The implementation was not optimized for performance. For example, we did not determine a minimum value for various time-outs. It is highly likely that with optimizations, the average topology formation time for the deployment scenario presented above will be lower than 78 seconds. Optimization of our prototype implementation and a more thorough evaluation of the optimized implementation is an ongoing work.

## 6 Conclusions

The idea of an arbitrary WSN but with regular topology backbone is quite promising since it provides a rich design space for simple, yet very efficient, network protocols, including those for real-time communications. Our distributed algorithm, its implementation and testing provide the evidence that hexagonal topology WSN is realizable in practice.

## 7 Acknowledgment

We thank Jianxin Chen for his help with the field tests. We also thank one of the anonymous reviewers for very valuable suggestions that improved the quality of this paper's presentation.

## References

- [1] E. Arıkan. Some complexity results about packet radio networks. *IEEE Transactions on Information Theory*, 30(4):681–685, Jul 1984.
- [2] J. Aspnes, T. Eren, D. K. Goldenberg, A. S. Morse, W. Whiteley, Y. R. Yang, B. D. Anderson, and P. N. Belhumeur. A theory of network localization. *IEEE Transactions on Mobile Computing*, 5(12):1663–1678, 2006.
- [3] D. M. Blough, M. Leoncini, G. Resta, and P. Santi. The K-Neigh protocol for symmetric topology control in ad hoc networks. In *MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing*, pages 141–152, New York, NY, USA, 2003. ACM.
- [4] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris. SPAN: An energy-efficient coordination algorithm for topology maintenance in ad hoc wireless networks. In *ACM Wireless Networks Journal*, pages 85–96, 2001.
- [5] M.-S. Chen, K. G. Shin, and D. D. Kandlur. Addressing, routing, and broadcasting in hexagonal mesh multi-processors. *IEEE Transactions on Computers*, 39(1):10–18, 1990.
- [6] M. R. Garey and D. S. Johnson. *Computers and Intractability — A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, 1990.
- [7] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 43(4):499–518, Nov. 2003.
- [8] D. Moss and P. Lewis. TEP 127: Packet link layer. <http://www.tinyos.net/tinyos-2.x/doc/html/tep127.html>.
- [9] S. Narayanaswamy, V. Kawadia, R. S. Sreenivas, and P. R. Kumar. Power control in ad-hoc networks: Theory, architecture, algorithm and implementation of the COMPOW protocol. In *European Wireless Conference*, pages 156–162, 2002.
- [10] K. S. Prabh and T. Abdelzaher. On scheduling and real-time capacity of hexagonal wireless sensor networks. In *ECRTS '07: Proc. of the 19th Euromicro Conference on Real-Time Systems*, pages 136–145. IEEE Press, Los Alamitos, CA, 2007.
- [11] R. Rajaraman. Topology control and routing in ad hoc networks: a survey. *SIGACT News*, 33(2):60–73, 2002.
- [12] V. Rajendran, K. Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient collision-free medium access control for wireless sensor networks. In *SenSys '03: Proc. of the 1st international conference on Embedded networked sensor systems*, pages 181–192, New York, NY, USA, 2003. ACM Press.
- [13] S. Ramanathan. A unified framework and algorithm for channel assignment in wireless networks. *Wirel. Netw.*, 5(2):81–94, March 1999.
- [14] I. Rhee, A. Warriier, M. Aia, and J. Min. Z-MAC: a hybrid MAC for wireless sensor networks. In *SenSys '05: Proc. of the 3rd international conference on Embedded networked sensor systems*, pages 90–101, New York, NY, USA, 2005. ACM Press.
- [15] P. Santi. Topology control in wireless ad-hoc and sensor networks. *ACM Comput. Surv.*, 37(2):164–194, 2005.
- [16] K. G. Shin. HARTS: A distributed real-time architecture. *Computer*, 24(5):25–35, 1991.
- [17] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *IEEE Transactions On Communications*, COM-32(3):246–257, 1984.
- [18] F. Xue and P. Kumar. The number of neighbors needed for connectivity of wireless networks. *Wireless Networks*, 10:169–181, 2004.