k-neigh alghoritm

In previous techniques the topology construction algorithm requires extra information

from the neighbor nodes other than their own presence, such as accurate Cartesian

coordinate (bi- or tri-dimensional location) or polar coordinate (distance and

angle). However, localization information is not always available or it could be very expensive to obtain. For example, location from GPS-enabled nodes can only be obtained in places where there is direct access to the satellite signals. Other localization techniques, like ultrasonic or ultrawide band-based, not only need a localization

protocol on top of the topology construction protocol, but could also increase

the communications overhead, as their range is very small compared with the radio

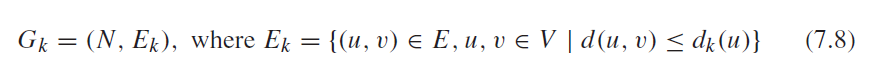
coverage. In the case of polar coordinates, the use of directional antennae increases

the price and complexity of the wireless devices.

Neighbor-based techniques overcome these problems, as they assume that nodes

only need to have the ability to determine the amount of neighbors, change their transmission power and, in some cases, calculate the distance between nodes. The main idea of these algorithms is to produce a connected topology by connecting each node with the smallest necessary set of neighbors, and with the minimum transmission power possible. Given that the nodes do not posses accurate location information, their decisions depend mostly on the probability of selecting the appropriate neighbors, the ones that would extend the network as far as possible. Under the assumption that the nodes are either uniformly or Poisson distributed, some properties have been found in connected topologies that define a bounded minimum appropriate size of the neighborhoods of a single node that *w.h.p.* would create a connected topology. As a result, most neighbor-based protocols for topology construction are based on the creation of a **K-neighbor graph**.

A K-neighbor graph is formally defined as in Equation 7.8 below:



where *dk(u)* is the distance from the *k*th closest neighbor of *u*. In other words, each node will adapt its transmission range in order to have a direct link to its closest *k* neighbors. The main question of this approach is how to select the optimal *k* such that it will produce a connected topology.

The definition of the minimum number of neighbors *k* that each node must have in order to preserve connectivity has been a well-studied problem. Most commonly used numbers for this parameter are between 6 and 8, or an average of 3 neighbors, as presented in [65, 85, 51]. Using the experiments presented in Section 7.2 to determine the giant component and the average node degree, we found the following practical results:

• In order to have a connected topology with probability 1, the topology with 10 nodes showed an average neighborhood of 7, and the topologies with 100 and1000 nodes showed an average neighborhood of 20 and 27, respectively.

• To have a connected topology with 90% probability, the topology with 10 nodes showed an average neighborhood of 5, the topology with 100 nodes showed an average neighborhood of 10, and the topology 1000 nodes showed an average neighborhood of 9.

***Modeling Energy Consumption***

In WSNs,energy consumption is one of the most important issues because each sensor node has a limited energy level. If all nodes become involved in sensing, redundancy will be increased and this will lead to consumption of unnecessary energy. Thus, it is fundamental to model the node energy consumption accurately. The consumed energy in sensors includes the energy required for sensing, receiving, transmitting and processing of data. The total consumed energy is usually dominated by the required energy for data transmission.Two cases may be considered

for the transmission mode of the nodes in the network. In the first case, nodes transmit with a fixed transmission power. This usually results in a fixed transmission range.

In the second case, nodes use a mechanism to adjust transmission power based on their distance to the next hop or sink. Required energy for a packet transmission in sensor i

can be modeled as 

where α represents the path loss exponent. The typical value of α for WSNs.

**A3 alghoritm**

The **A3 protocol** proposed in [128] is an example of a distributed implementation of a growing a tree algorithm. A3 builds a non-optimal connected dominating set over an originally connected graph considering the remaining energy in the nodes and the distance between them. The tree is built using four types of messages: *Hello*

*Message*, *Parent Recognition Message*, *Children Recognition Message*, and *Sleeping Message*. The CDS building process is started by a predefined node that might be the sink, right after the nodes are deployed. The sink, node A in Figure 8.3a, starts the protocol by sending an initial *Hello Message*. This message will allow

the neighbors of A to know their “parent”. In Figure 8.3a, nodes B, C, D, and E

will receive the message. Nodes F and G are out of reach from node A. If the node

that receives the message has not been covered by another node, it sets its state as

covered, adopts the sender as its “parent node”, and answers back with a *Parent*

*Recognition Message*, as shown in Figure 8.3b. This message also includes a selection

metric (explained later) that is calculated based on the signal strength of the

received *Hello Message* and the remaining energy in the node. The metric will be

used later by the parent node to sort the candidates. If the receiver has been already

covered by another node, it ignores the *Hello Message*.

The parent node waits a certain amount of time to receive the answers from its

neighbors. Each answer (metric) is stored in a list of candidates. Once this timeout

expires, the parent node sorts the list in decreasing order according to the selection

metric. The parent node then broadcasts a *Children Recognition Message* that includes

the complete sorted list to all its candidates. In Figure 8.3c, node A sends

the sorted list to nodes B, C, D, and E. Once the candidate nodes receive the list,

they set a timeout period proportional to their position on the candidate list. During

that time nodes wait for *Sleeping Messages* from their brothers. If a node receives a

*Sleeping Message* during the timeout period, it turns itself off, meaning that one of

its brothers is better qualified to become part of the tree. Based on this scheme, the

best node according to the metric will send a *Sleeping Messages* first, blocking any

other node in its range. Therefore, only the other candidate nodes outside its area of

coverage have the opportunity to start their own generation process. For example, in

Figure 8.3d, node D received a *Sleeping Message* from E before its timer expired, so

it turned off. Otherwise, it sends a *Sleeping Message* to turns its brothers off. At that

time, this particular node becomes a new “parent node” and starts its own process of

looking for candidates. Finally, if a parent node does not receive any *Parent Recognition Message* from its neighbors, it also turns off, such as the case of nodes E and

B in the final topology, as shown in Figure 8.3e, given that they have no children.

What are geometric graphs?

• The vertices of the graphs are geometric objects.

• The edges are placed based on a geometric relationship

between the objects.

Geometric graphs: More examples

• Vertices: Line segments in Rd. Edges: Between two line

segments that intersect.

• Vertices: Voronoi cells of a point set in Rd. Edges: Between

two cells that share a d − 1 dimensional facet.

• Vertices: Points in Rd. Edges: From each point to the k points

closest to it.

What are random graphs?

Given a graph G = (V,E), a random graph is a probability

distribution over the set of all subgraphs of G.

What are geometric random graphs?

Given a geometric graph G = (V,E) a geometric random graph is a

probability distribution over the set of all subgraphs of G

Geometric random graphs: Some examples

• The unit disk graph on a randomly distributed set of points.

• The Voronoi graph with each Voronoi cell retained in the graph

independently with probability p and removed with probability

1 − p.

**Yao graph (yg)**

The algorithm presented in [139], called **Yao Graph (YG)**, is an early version

of a direction-based topology control technique in multidimensional spaces. This

algorithm works in two phases: (1) reduce the original graph into a subgraph that

contains a MST, and then (2) remove the extra edges that will leave only the MST.

The subgraph produced after the first stage of the process is called a Yao Graph.

Assume that from each node *v* in *V* , there are k>=6 equally distant rays originating

from *v* forming *k* cones. The selection of the number of cones determines

that each cone has at most an angle *θ* ≤ 60◦ or *θ* ≤ *π/*3, which guarantees not only

that each node will have at most *k* neighbors, but that any distance between nodes

inside the cone is not the longest edge of the triangle formed by the node in the

center and both its neighbor nodes. This characteristic is useful for the construction

of the MST. Figure 7.11 shows an example of the Yao Graph of a small MaxPower

topology. After the cones have been established, node *v* surveys its neighbors on each one

of the *k* cones. Node *v* calculates the distances from all the neighbors and selects

the shortest edge from each cone, creating an undirected edge *(v, u)*, where node

*u* is the closest node to *v* in that cone. If there is any kind of tie in the closest

neighbors selection, any methodology like random selection or using the node ID can be used to break the tie. The authors prove that a MST is contained in the set of

edges resultant from this first stage. In the second phase, in order to calculate the MST, a global and centralized

processor is needed to solve the post-office problem, which determines, for each

point uV the closest point u such that *d(v, u) < d(v,w)*, ∀*w* ∈ *V* , *w* \_= *u*. The complete MST algorithm has a complexity of 8 .*f (n)* + *O(n log log n)*, where

*f (n)* is the cost of finding the shortest edge on a cone.

However,

it is important to mention that this algorithm does not guarantee connectivity if

the transmission range of the selected neighbor is not greater than the transmission

range of the evaluating node. An extension of the Yao Graph that overcomes the

heterogeneous communication range problem is presented in [74].

The YG protocol is shown to have the following nice properties:

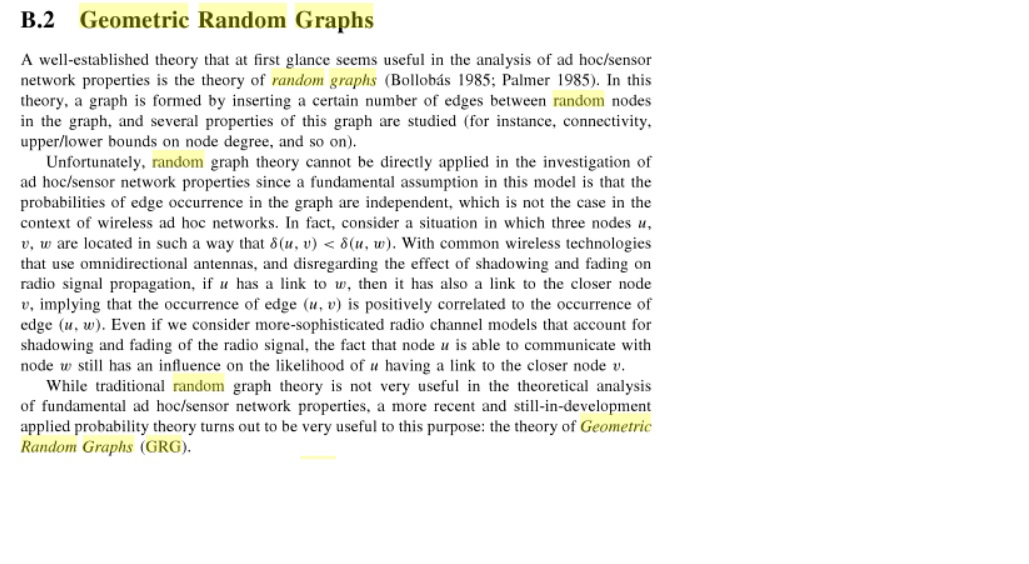
• Bounded node degree equal or less than *k*, where *k* ≥ 6 cones.

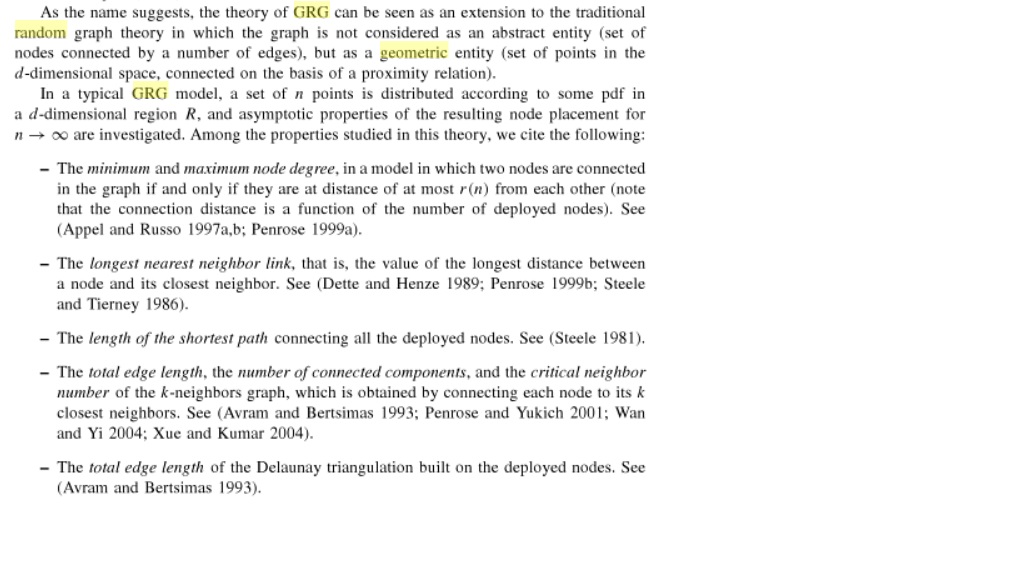
• Network connectivity is preserved, as long as nodes are homogeneous.

• Distributed (Yao Graph) and centralized (MST) protocol.

• Local information, at least during the first phase.

Graph GRG





In this research, a new distributed topology control technique is presented that enhances energy efficiency and

reduces radio interference in wireless sensor networks. Each node in the network makes local decisions about its transmission power and the culmination of these local decisions produces a network topology that preserves global connectivity. Central to this topology control technique is the novel Smart Boundary Yao Gabriel Graph (SBYaoGG) and optimizations to ensure that all links in the network are symmetric and energy efficient. Simulation results are presented demonstrating the effectiveness of this new technique as compared to other approaches to topology control.

III. SMART BOUNDARY YAO GABRIEL GRAPH (SBYAOGG)

*A. Objectives*

There were two design objectives in developing the envisaged topology control technique for WSNs. The first objective was that it should be energy efficient and the second was that it should have low interference. Performance measures were used to determine how well these objectives were met. The

relative performance of the new technique compared to other well-known approaches to topology control was also used to evaluate how well these objectives were met.

These objectives, however, are competing as improving energy efficiency, as measured through the energy stretch factor of a generated topology, increases the level of interference in the network as measured by the maximum and average node degree of the generated topology. In addition to this, topology control involves a compromise between energy conservation and network connectivity. The topology control technique that was developed was designed to meet the two competing objectives given above by finding a balance between them using certain heuristics given in Section III-C2.

*B. Requirements*

To meet the set out design objectives, the routing subgraph T produced by the topology control technique from the original graph G had to meet certain requirements. A number of the requirements were for minimum energy unicast; these are the following.

1) Constant power stretch factor, i.e., the graph should be a power spanner of G .

2) Linear number of edges, i.e., the graph must be sparse.

3) Easy computation in a distributed and localized way.

In addition to this, the subgraph T had to be:

1) Connected with high probability if the original graph G is connected.

2) Planar, meaning that no two edges in the graph cross each other. This will enable some localized routing algorithms to work with the generated topology such as Greedy Face Routing (GFR), Greedy Perimeter Stateless Routing (GPSR), Adaptive Face Routing (AFR), and Greedy Other

Adaptive Face Routing (GOAFR) [21].

*C. Yao-Gabriel Graph With Smart Boundaries*

In order to develop a technique that produced a network topology that met the objectives that have been set out and that adhered to the identified requirements and conformed to the choices made, it was decided to create a graph algorithm that is a hybrid of different proximity graph algorithms. The algorithm is a mixture of the Gabriel graph algorithm and the Yao graph algorithm, with the use of smart region boundaries. The algorithm is referred to as the Smart Boundary Yao Gabriel Graph (SBYaoGG). The topology is generated by first computing the Gabriel graph from the Unit Disk Graph (UDG) at maximum transmitter power and then computing the Yao graph on the reduced topology to produce the final topology. By computing the Gabriel graph from the UDG some of the requirements for the final topology are met.

• The graph produced is planar.

• The graph has a linear number of edges.

After computing the Yao graph from the Gabriel graph some other requirements for the final topology are met.

• The graph is connected. This is because both the Gabriel graph and the Yao graph are connected if the original graph is connected.

• The graph is a power spanner of the original UDG. This is because both the Yao graph and the Gabriel graph are

power spanners.

In addition to this:

• The final graph is easily computable in a distributed and localized way. The algorithm uses node position information which can be obtained from neighbor nodes. The edges are then computed locally by using only the neighbor position information. Therefore, all of the requirements for the topology control technique were met. This contributed to meeting the objectives of the final graph in that it is energy efficient and has low interference. A low physical node degree necessary for minimal interference and a low-power stretch factor necessary for high

energy efficiency are two opposing goals, as has been noted. *1) Pruning the Edges of the Gabriel Graph:* The Gabriel graph computed on the UDG has its edges pruned by computing the Yao graph on the reduced topology. This, in effect, generates the previously developed YaoGabriel graph. In order to achieve low interference, it is desirable to reduce the node degree as much as possible, while maintaining the power spanner properties

of the Gabriel graph. The YaoGabriel graph can achieve this. However, further reduction in interference levels can be obtained by variable selection of the axes of the cones for each region of the Yao graph. The procedure employed to reduce interference was as follows.

• Prune the edges of the Gabriel graph using the Yao graph.

• Use large regions in computing the Yao graph.

• Select the axes of cones for each region of the Yao graph using heuristics.

• Reduce the transmitter power of each node to the lowest level so that it allows it to reach its furthest neighbor in the final topology.

*2) Determining the Region Boundaries of the Yao Graph:*

A heuristic that was used whilst forming the reduced topology graph was to align the axis of the first cone used in the Yao graph computation to the region where nodes are most densely deployed. This can be accomplished by obtaining the unit direction vectors of all the neighboring nodes and then calculating the average direction vector.

The average direction vector is then used as the axis of the cone for the first Yao graph region. The neighbor direction vectors and the average direction vector are illustrated in Fig. 1.

The cones of the Yao graph are as shown in Fig. 2. In the case

of Figs. 4 and 5, corresponding to a Yao graph with

three cones. It can be seen that aligning the axis of one of the

cones to the average direction vector, results in a cone where a

high number of neighbor nodes fall into. It is possible that in

certain arrangements all the neighbors will fall into this cone,

which will mean that the number of edges calculated during

topology control will be reduced. topology control will be reduced.

An option was to use the neighbor centroid, for the following

reason: Opposed to the average direction vector, the centroid has

the drawback that neighbors that are further away will have coordinates

that dominate over closer neighbor coordinates. The

average of unit direction vectors overcomes this drawback. Selection of is necessary to maintain connectivity. The largest value of was used in simulations

when setting the boundaries of the Yao graph, in order to reduce

the node degree as much as possible, while maintaining

connectivity.

*3) Optimizations:* Once the edges of the Gabriel graph are

pruned, there will be some asymmetric edges. One optimization

that was used was to make all the edges symmetric by adding

the reverse edge of any asymmetric link where u ,v but v,u . If a node has a link to a node , but node does not have a link to node u , then node v adds a link to node u

. This reduces the power stretch factor of the final graph and ensures

that there are no asymmetric links which can pose a burden to a

MAC protocol.

*4) Algorithm:* The following algorithm describes how to

construct the SBYaoGG, detailing the node in the network goes

through.

**Algorithm**: Construction of SBYaoGG

1. The node discovers its neighbor nodes by broadcasting

at maximum power.

2. The Gabriel graph is constructed locally.

3. The unit direction vectors of neighbor nodes in the Gabriel

graph are computed.

4. The average direction vector is computed.

5. The axis of the cone of the first region to use in computing

the Yao graph is set to correspond to the average direction

vector.

6. The Yao graph is computed from the Gabriel graph,

producing the reduced topology.

The final step in obtaining the SBYaoGG is to optimize the

reduced topology in order to ensure low interference and good

power spanner properties. Two optimizations were made.

1) All edges are made symmetric by adding the reverse edge for any asymmetric link.

2) Transmitter power levels are set to the lowest level that will allow each node to reach all the nodes with which it has an edge.

After this the SBYaoGG is fully formed and can be used as input to a routing algorithm.